Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra systémového inženýrství

Diplomová práce

Uživatelské testování softwaru a projekty ICT ve společnosti Sazka a.s.

Lenka Slišková

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Lenka Slišková

Systémové inženýrství a informatika
Projektové řízení

Název práce

Uživatelské testování softwaru a projekty ICT ve společnosti Sazka a.s.

Název anglicky

User SW testing and ICT projects in the company Sazka a.s.

Cíle práce

Cílem práce je na základě popisu a rozboru současné praxe uživatelského testování v projektech ICT ve společnosti Sazka a.s. identifikovat slabá místa, nedostatky a navrhnout jejich možná řešení, a to v kontextu soudobé praxe.

Metodika

Metodika: Po studiu vybrané literatury bude proveden sběr a rozbor dat a poznatků o současném stavu uživatelském testování a řízení ICT projektů ve vybrané firmě. Na základě provedeného rozboru bude vypracován vlastní návrh zlepšení, tj. s využitím procesních, kompetenčních a jiných modelů bude zachycen současný stav, provedeno zobrazení, identifikována slabá místa a proveden návrh konceptuálního modelu (navrženy nástroje). Vlastní výsledky budou diskutovány v praxi dané firmy s projektovým týmem. Po vypracování praktické části bude sepsána část teoretická.

Harmonogram

1. Studium vybrané literatury
2. Sběr dat a poznatků v praxi firmy
3. Identifikace slabých míst a nedostatků
4. Tvorba konceptuálního modelu a nových nástrojů
5. Návrh doporučení pro zlepšení stavu
6. Diskuze výsledků s projektovým týmem
7. Vypracování literární rešerše

Doporučený rozsah práce

60 – 70 stran

Klíčová slova

Projektové řízení, tvorba softwaru, uživatelské testování, vývojářský tým, projektový tým, vývoj SW.

Doporučené zdroje informací

BUREŠ, Miroslav, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ. Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu. Praha: Grada, 2016. Profesionál. ISBN 978-80-247-5594-6.

KERZNER, H. *Project management : a systems approach to planning, scheduling, and controlling.* Hoboken: John Wiley & Sons, 2013. ISBN 978-1-118-02227-6.

PATTON, Ron a Anna HAVLÍČKOVÁ. Testování softwaru: průvodce testováním. Praha: Computer Press, 2002. Programování. ISBN 80-722-6636-5.

PROJECT MANAGEMENT INSTITUTE. *The standard for program management.* Newtown Square: Project Management Institut, 2013. ISBN 978-1-935589-68-6.

ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. Řízení kvality softwaru: průvodce testováním. Brno: Computer Press, 2013. ISBN 978-80-251-3816-8.

SCHWALBE, Kathy, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ. Řízení projektů v IT: klíčové otázky pro efektivitu testovacího procesu. Brno: Computer Press, 2007. Kompletní průvodce (Computer Press). ISBN 978-80-251-1526-8.

SVOZILOVÁ, A. *Projektový management : systémový přístup k řízení projektů.* Praha: Grada, 2016. ISBN 978-80-271-0075-0.

SVOZILOVÁ, A. *Zlepšování podnikových procesů.* Praha: Grada, 2011. ISBN 978-80-247-3938-0.

Předběžný termín obhajoby

2019/20 LS – PEF

Vedoucí práce

doc. Ing. Jan Bartoška, Ph.D.

Garantující pracoviště

Katedra systémového inženýrství

Elektronicky schváleno dne 17. 3. 2020

doc. Ing. Tomáš Šubrt, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 17. 3. 2020

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 04. 04. 2020

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Uživatelské testování softwaru a projekty ICT ve společnosti Sazka a.s." jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 5.4.2020 _____

Poděkování

Ráda bych touto cestou poděkovala svému vedoucímu práce, panu docentu Bartoškovi za užitečné rady a vedení diplomové práce. Dále pak rodině za podporu při psaní práce. Nakonec bych ráda poděkovala společnosti Sazka a.s. a to především zástupcům testovacího oddělení za poskytnutá data a spolupráci

Uživatelské testování softwaru a projekty ICT ve společnosti Sazka a.s.

Abstrakt

Diplomová práce je zaměřena na problematiku uživatelského testování softwaru jakožto realizační fáze projektů ve společnosti Sazka a.s.

Teoretická část je zpracována na základě studia odborné literatury, která se zabývá testováním. Vysvětluje základní pojmy z oblasti testování, typy prováděných testů rozdělených dle různých hledisek, životní cyklus testování softwaru a metodiku vývoje softwaru, která je používaná v popisované společnosti.

V praktické části je nejdříve představena společnost Sazka a.s. Je popsán vývoj softwaru a stav, ve kterém se aktuálně nachází uživatelské testování v této společnosti. Dále jsou identifikována slabá místa, která se v rámci uživatelského testování vyskytují. Na základě zhodnocení aktuálního stavu testování a identifikace slabých míst byl vytvořen návrh na zlepšení současné situace.

Jedná se především o zavedení školení pro nové test exekutory a vytvoření školicího manuálu. Dále pak vytvoření sdílených dokumentů, one to one meetingy, zavedení informační tabule a nákup nových zařízení.

Klíčová slova: Projektové řízení, tvorba softwaru, uživatelské testování, vývojářský tým, projektový tým, vývoj SW.

User SW testing and ICT projects in the company Sazka a.s.

Abstract

This thesis is focused on the issue of the user software testing as the implementation phase of projects in the company Sazka a.s.

The theoretical part is based on the study of literature, which deals with testing. It explains the basic concepts of testing, the types of tests performed according to different aspects, the software testing life cycle and the software development methodology used in the described company.

In the practical part, the Sazka a.s. company is introduced. It describes the software development and the current state of the user testing in this company. Furthermore, weaknesses that occur within user testing are identified. Based on the evaluation of the current state of testing and identification of weak points, a proposal is made to improve the current situation.

These include the introduction of training for new test executors and the creation of a training manual. Furthermore, the creation of shared documents, one to one meeting, the introduction of an information board and the purchase of new equipment.

Keywords: Project management, software development, user testing, development team, project team, SW development.

Obsah

1 Úvod **12**

2 Cíl práce a metodika **14**

2.1 Cíl práce 14

2.2 Metodika 14

3 Teoretická východiska **15**

3.1 Co je to testování 15

3.1.1 Vymezení pojmu defekt 17

3.2 Základní rozdělení testů 18

3.2.1 Statické a dynamické testování 19

3.2.2 Black-Box a White-Box testování 20

3.2.3 Manuální a automatizované testování 21

3.3 Typy testů podle úrovně vývoje 22

3.3.1 Testování jednotek – Unit testy 22

3.3.2 Integrační testování 23

3.3.3 Systémové testování 23

3.3.4 Akceptační testování 24

3.4 Další používané testy 24

3.4.1 Smoke testy 25

3.4.2 Regresní testy 25

3.4.3 Konfirmační testování 26

3.5 Dokumentace v testování 26

3.5.1 Testovací plán 26

3.5.2 Test Case 27

3.5.3 Testovací scénář 28

3.5.4 Report defektu 29

3.6 Testovací tým 32

3.6.1 Motivace testerů 33

3.7 Vývoj softwaru – model Waterfall 33

3.8 Životní cyklus testování softwaru 36

4 Vlastní práce **39**

4.1 Charakteristika společnosti 39

4.1.1 Organizační struktura společnosti 40

4.2 Vývoj softwaru a testování 41

4.2.1 Vývoj her 42

4.2.2 Vývoj aplikací 43

4.2.3 Průběh uživatelského testování ... 44
4.2.4 Životní cyklus incidentu ... 45

5 Současný stav testování ... 47
5.1 Testované platformy ... 47
5.1.1 Webová stránka ... 48
5.1.2 Mobilní aplikace ... 48
5.1.3 POS terminály ... 50
5.2 Testovací zařízení ... 51
5.3 Velikost a složení týmu – organizační struktura ... 52
5.4 Plánování testování ... 54
5.5 Software SpiraTest ... 55
5.5.1 Způsob testování ve SpiraTest ... 55
5.5.2 Incidenty ... 56
5.6 Typy prováděných testů ... 56
5.6.1 Smoke testy ... 57
5.6.2 Funkční testy ... 57
5.6.3 Testování dokumentace – Gameplan ... 58
5.6.4 Konfirmační testování – retestování ... 58
5.6.5 Regresní testování ... 58

6 Identifikace slabých míst spojených s uživatelským testováním ... 59
6.1 Absence školení ... 59
6.2 Zápis incidentů ... 59
6.3 Absence testovacího plánu ... 61
6.4 Jednotné testovací scénáře ... 61
6.5 Nedostatek testovacích zařízení ... 61
6.6 Výkonnost testerů ... 62
6.7 Nedostatečná komunikace a informovanost ... 62

7 Návrh na změnu ... 63
7.1 Důkladné zaškolení a školicí manuál ... 63
7.1.1 Školení testerů ... 64
7.1.2 Školicí manuál ... 65
7.2 Vytvoření sdíleného dokumentu ... 67
7.3 Nákup a přesun testovacích zařízení ... 68
7.4 One to one meeting ... 68
7.5 Informační tabule ... 69

8 Závěr ... 71

9 Seznam použitých zdrojů ... 73

Seznam obrázků

Obrázek 1 Vztah mezi fází, ve které je defekt identifikován, a cenou za jeho odstranění ..17
Obrázek 2 V – model ..22
Obrázek 3 Příklad životního cyklu defektu ..31
Obrázek 4 Životní cyklus testování softwaru ..36
Obrázek 5 Certifikáty získané společností Sazka a.s. ..40
Obrázek 6 Vývoj her z pohledu Sazky ..43
Obrázek 7 Vývoj aplikací ..44
Obrázek 8 Průběh uživatelského testování ..45
Obrázek 9 Životní cyklus incidentu ..46
Obrázek 10 Náhled testovací webové stránky Sazka.cz ..48
Obrázek 11 Náhled aplikace Sazka ..49
Obrázek 12 Náhled aplikace SazkaBet ..50
Obrázek 13 Organizační struktura testovacího oddělení ..52
Obrázek 14 Hlavní obrazovka softwaru SpiraTest ..55
Obrázek 15 Formulář pro vypsání nového incidentu ..60

Seznam tabulek

Tabulka 1 Příklad šablony pro vytvoření test casu ..28
Tabulka 2 Příklad tvorby testovacího scénáře ..29
Tabulka 3 Přehled testovaných objektů na platformách ..47
Tabulka 4 Soupis testovacích zařízení ..51
Tabulka 5 Slabé stránky a návrhy jejich řešení ..63
Tabulka 6 Návrh harmonogramu školení ..64
Tabulka 7 Obsah školicího manuálu ..66
Tabulka 8 Návrh formátu sdíleného dokumentu ..68

Seznam použitých zkratek

CSR – Customer Social Responsibility

FTS – Finanční a Transakční Služby

GUI – Graphic User Interface

ICT – Information and Communication Technologies

POS – Point Of Sale

QA – Quality Assurance

SIT – System Integration Tests

UAT – User Acceptance Testing

1 Úvod

Jako téma diplomové práce jsem si zvolila uživatelské testování softwaru a projekty ICT ve společnosti Sazka a.s.

Testování softwaru patří do realizační fáze řízení projektu a je součástí řízení kvality. V projektech však často nad kvalitou vítězí nutnost dodržet termín nebo rozpočet. Splnění lhůty dodání na úkor kvality se však může vymstít stejně jako přesun financí z ověření kvality do jiné oblasti. Toto platí obecně. Někdy je oprava chyb jednoduchá a rychlá, ale u ICT projektů mohou být důsledky fatální.

Byť je uživatelské testování velice podstatnou součástí vývoje kvalitního softwaru, v praxi se mnohdy ukazuje jako proces, který je drahý, vyžaduje velké úsilí a není zcela spolehlivý v tom smyslu, že ne vždy se podaří odhalit všechny defekty. Možná i proto je to část vývoje, která je často zanedbávaná, opomíjená a oproti jiným oblastem vývoje zdánlivě nedůležitá. Často se na ní šetří a není na ni kladen velký důraz. Přitom právě testování je důležité pro vývoj kvalitního, zcela funkčního produktu. Je možné, že i produkt, na který bylo vynaloženo značné množství prostředků se v důsledku neodhaleného defektu stane pro koncového uživatele nepoužitelným nebo nebude využit celý jeho potenciál. Přitom například použitím správné metodiky testování, dobrým plánováním, řízením a prováděním testů se testování stává velice výhodným.

S postupující digitalizací roste potřeba vývoje softwaru, rozšiřují se týmy vývojářů a programátorů a proporcionálně k tomu by se měl zvyšovat i počet testerů.

V současné době si tuto skutečnost začíná uvědomovat stále více firem, které pak investují do rozvoje testovacího oddělení a zdokonalení testovacích procesů. To lze označit za pozitivní trend, který však vykazuje určité rezervy. Hlavní slabinou je postupné rozšiřování týmu bez odpovídajícího nastavení procesu komunikace a řízení.

K postupnému rozšiřování testovacího týmu dochází i ve společnosti Sazka a.s., kterou jsem si vybrala pro zpracování tématu diplomové práce. Za poslední dva roky podstoupila značný rozvoj a rozšíření testovacího oddělení a tento rozvoj ještě stále pokračuje. I když prošlo testovací oddělení změnami, má ještě neustále v některých místech rezervy. Tato práce se zaměří pouze na oblast uživatelského testování, kde vidím největší potenciál pro zlepšení.

Výběr této společnosti jsem zvolila z toho důvodu, že ve společnosti Sazka a.s. osobně pracuji. Konkrétně pracuji v oblasti uživatelského testování, a tudíž mohu dobře zhodnotit slabiny této oblasti a konzultovat je s kolegy.

V praktické části bude nejdříve představena společnost Sazka a.s., její organizační struktura a předmět činnosti. Bude nastíněn vývoj softwaru, průběh testování a životní cyklus incidentu. Dále bude popsán stav, ve kterém se aktuálně nachází uživatelské testování v této společnosti, což bude zahrnovat testované platformy, typy prováděných testů, organizační strukturu testovacího oddělení a popis softwaru používaného pro podporu testování. Poté budou identifikována slabá místa, která se v rámci uživatelského testování vyskytují. Na základě zhodnocení aktuálního stavu testování a identifikace slabých míst bude vytvořen návrh na zlepšení současné situace.

Hlavním přínosem práce bude právě vytvoření návrhu na odstranění identifikovaných slabých míst v procesu uživatelského testování.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem práce je na základě popisu a rozboru současné praxe uživatelského testování v projektech ICT ve společnosti Sazka a.s. identifikovat slabá místa, nedostatky a navrhnout jejich možná řešení, a to v kontextu soudobé praxe.

2.2 Metodika

Po studiu vybrané literatury bude proveden sběr a rozbor dat a poznatků o současném stavu uživatelském testování a řízení ICT projektů ve vybrané firmě. Na základě provedeného rozboru bude vypracován vlastní návrh zlepšení, tj. s využitím procesních, kompetenčních a jiných modelů bude zachycen současný stav, provedeno zobrazení, identifikována slabá místa a proveden návrh konceptuálního modelu (navrženy nástroje). Vlastní výsledky budou diskutovány v praxi dané firmy s projektovým týmem. Po vypracování praktické části bude sepsána část teoretická.

3 Teoretická východiska

3.1 Co je to testování

Definic testování existuje mnoho a některé si i navzájem odporují. Každý autor i publikace zastává jiný pohled na to, co je to testování a co k němu naopak nepatří, ale není jediná správná definice.

Roudenský a Havlíčková definují testování jako „proces řízeného spouštění softwarového produktu s cílem zjistit, zda splňuje specifikované či implicitní potřeby uživatelů“ [ROUDENSKÝ, HAVLÍČKOVÁ, 2013, s. 45].

Myers (2012) chápe testování jako spouštění programu nebo systému za účelem hledání chyb [MYERS, 2012].

V glosáři ISTQB je popsáno jako „proces, který se skládá ze všech aktivit životního cyklu (dynamických i statických). Týká se plánování, přípravy a hodnocení softwarových produktů a souvisejících pracovních produktů s cílem určit, zda splňují specifikované požadavky, ukázat, že vyhovují účelu a najít defekty“ [ISTQB GLOSSARY, 2018, s. 33].

I když je definic mnoho, účel testování je vcelku jasný. Jedná se o ověření a měření kvality softwaru, tedy míry naplnění požadavků na něj, ne o pouhé hledání chyb. Chyby mohou být různé, a ne všechny je žádoucí je opravovat. Může se jednat o chyby, které jsou zanedbatelné, nemají vliv na funkci softwaru a jejich náprava by byla velice náročná. Takové nalezené chyby tak můžeme považovat za vedlejší cíl [ROUDENSKÝ, HAVLÍČKOVÁ, 2013].

Klíčovým prvkem při testování je testovací tým. Každý člen týmu by měl testovanému produktu a požadavkům klienta rozumět, měl by být precizní a mít dobré logické uvažování. Je nutné stanovit i záběr testování, výběr testů, sbírání dat a příprava nástrojů, které jsou potřeba k testování. Také se kontrolují požadavky na produkt, jestli jsou v takové kondici, aby bylo možné jednoznačně zkontrolovat jejich plnění.

Většinou, pokud se na projektu potřebuje někde ušetřit, šetří se na testování. Jak ušetřit na testování? Zkrátit jeho dobu, najmout jako testery brigádníky a dočasné pracovníky, omezit techniky testování. Spousta projektových manažerů si neuvědomuje, že při zkrácení etapy testování ve fázi vývoje sice dočasně sníží náklady na projekt, ale v dlouhodobém hledisku se jim to vrátí zpět při provozování a údržbě software. Produkt je nutné upravovat a opravovat chyby zjištěné až za provozu, opravovat data, která byla poškozena,

což ve většině případů stojí mnohem více než ušetřené náklady na testování. K takovým pozdějším opravám by vůbec nemuselo dojít a chyby by byly včas nalezeny [BUREŠ, RENDA, DOLEŽEL, aj., 2016].

Axiomy testování

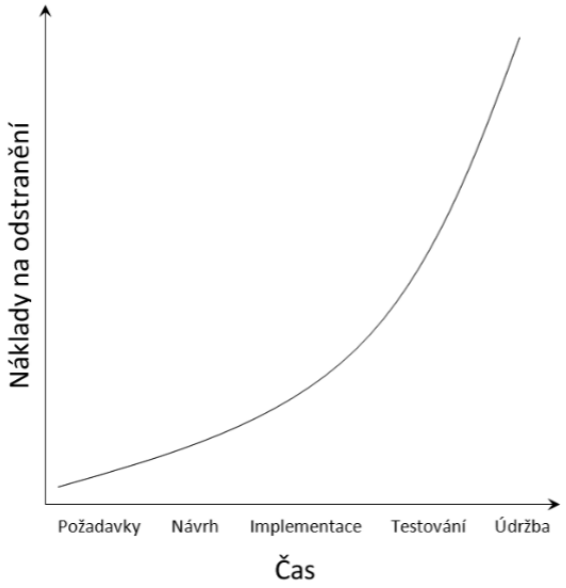
I přes značnou pečlivost testerů nelze říct, že provedené testy zajistí stoprocentní bezchybnost. S testováním jsou spojeny určité axiomy, které popisují nedostatky spjaté s testováním. Tyto axiomy popsal Ron Patton následovně:

- Není možné otestovat kompletně všechny možné případy zadání.
- Je velmi složité odhalit, že bylo ve specifikaci na něco zapomenuto.
- Specifikace není vždy jednoznačná nebo dokonce chybí.
- Testování bývá podceňováno nebo špatně pochopeno.
- Výsledky testování mohou být ovlivněny špatnou komunikací v týmu nebo se zákazníkem.
- Testování musí být přizpůsobeno testovanému produktu, není možné stejným způsobem otestovat různý software.
- V průběhu testování se mohou stát některé testy neefektivními. Testy je nutné neustále přizpůsobovat měnícím se okolnostem.
- Dvě různé chyby se mohou navenek projevovat stejně, nebo jedna chyba se může navenek projevovat různě.
- Testování je citlivé na vstupní data, ta by měla být realistická. Vygenerování jednotvárných dat není ideální [PATTON, 2002].

Hodnota testování

Testování je fází projektu, které je často velice podceňováno. Jedná se totiž o prvek, u kterého se někdo může domnívat, že lze při vývoji produktu vynechat či omezit. Pravdou ale je, že nedostatečné testování může projekt výrazně prodražit, protože čím později je chyba odhalena, tím je její oprava náročnější a dražší, jak je zřejmé z Obrázku 1 [ROUDENSKÝ, HAVLÍČKOVÁ, 2013].

Obrázek 1 Vztah mezi fází, ve které je defekt identifikován, a cenou za jeho odstranění



Zdroj: ROUDENSKÝ, HAVLÍČKOVÁ, 2013

3.1.1 Vymezení pojmu defekt

Pojem defekt je v procesu testování velice důležitým prvkem. Často ale dochází k jeho zaměňování s jinými pojmy z důvodu překladu z angličtiny. Proto je důležité tento pojem přiblížit a vymezit. Nejasnosti vznikají především mezi pojmy bug, defekt, chyba, selhání a incident.

- **Defekt** – známý pod anglickým pojmem „bug“, je určitá odchylka ve způsobu fungování aktuální verze softwaru od očekávaného fungování či chování. Defekt je následkem vytvořené chyby.
- **Chyba** – může vytvořit například vývojář špatně zapsaným kódem nebo analytik špatně vytvořeným zadáním.
- **Selhání** – můžeme rozumět jako zhroucení systému nebo jen jeho části. Nastává právě kvůli nějakému defektu, který selhání způsobí.
- **Incident** – definován dle ISTQB jako „jakákoliv událost, která vyžaduje prozkoumání“. Tento pojem se ale v praxi může používat i pro vady identifikované na produkčním prostředí [BUREŠ, RENDA, DOLEŽEL, aj., 2016].

Aby se jednalo o defekt z pohledu specifikovaných požadavků musí být podle Pattona splněný alespoň jeden z následujících čtyř bodů a sice:

- Systém vykonává něco, co by podle specifikace vykonávat neměl.
- Systém nevykonává to, co by podle specifikace vykonávat měl.
- Systém vykonává něco, co není specifikováno.
- Systém vykonává něco, co specifikování není, ale mělo být.

Připojit k již zmíněným čtyřem bodům lze i případ, kdy je software těžko srozumitelný, uživatelsky nepřívětivý, anebo pomalý.
Jde tedy o rozpor mezi specifikovanými požadavky či oprávněnými očekáváními zákazníka a vytvořeným softwarem. Všechny tyto defekty mají svůj původ. Převážně se jedná o původ ve zdrojovém kódu, který tvoří zhruba 70-80 % defektů. Zbytek tvoří specifikace, návrh a další méně zastoupené kategorie [PATTON, 2002].

3.2 Základní rozdělení testů

Možností, jakými typy testů se dá testovat software, je celá řada a mohou se dělit podle několika hledisek, například zda:

- Test vyžaduje spuštění softwaru pro jeho otestování.
    - statické
    - dynamické
- Software je testován zvenku či zevnitř.
    - Black-Box
    - White-Box
    - Grey-Box
- Software je testován člověkem nebo je k jeho otestování použit program.
    - manuální
    - automatizované
    - exploratory

3.2.1 Statické a dynamické testování

Software můžete otestovat dvěma způsoby, a to dynamicky nebo staticky. Statický test, znamená to, že lze provádět test bez použití dotyčného softwaru. Dynamické testování představuje testování softwaru, který je spuštěn [SCHOTANUS, 2009].

Statické testování

Ve všech fázích procesu vývoje systému se vytváří jeden či více dokumentů, a to například požadavky, specifikace systému nebo technické specifikace. Dokumenty vytvořené ve všech fázích vývoje bývají testovány na:

- úplnost,
- správnost,
- relevantnost,
- jednoznačnost,
- konzistenci.

Toho se dosahuje čtením specifikací a hodnocením obsahu. Takové testování je považováno právě za statické, tedy bez spuštění vyvíjeného softwaru [BUREŠ, RENDA, DOLEŽEL, aj., 2016].

Statické testování je účinná a efektivní metoda prevence defektů především z toho důvodu, že dojde k odhalení případné chyby ještě před zahájením dalších činností a je třeba upravit jen dokument, což představuje nižší náklady na opravu než v případně odhalení defektu v pozdější fázi. Pokud je chyba objevena až při akceptačních testech, musí být upraven nejen samotný software, ale také test casy, dokumenty a návody k použití. Tato situace může nastat například z důvodu nesprávné interpretace dokumentu.

Dynamické testování

Principem dynamického testu je nutnost, aby software, jeho části nebo komponenty byly v provozu. Jeho cílem je zjistit, zda testované objekty splňují požadavky a očekávání. Posuzují se na základě vložených vstupů či spuštěných akcí a k nim příslušných výstupů.

Pro dynamické testování jsou mnohdy nezbytné určité nástroje, na kterých lze software testovat. V systémech s Graphic User Interface (GUI) neboli v překladu s grafickým uživatelským rozhraním se zadávají vstupy a spouští akce na obrazovce, zatímco v systémech, které nemají GUI se využívají nástroje pro databázi SQL (dotazovací jazyk).

3.2.2 Black-Box a White-Box testování

Další možností, jak testovat systém je testování zevnitř nebo zvenku. Testováním zvenku je myšleno testování funkčnosti systému, což je označováno jako Black-Box test. Naopak White-Box test, kterým je míněno testování zevnitř, představuje testování vnitřních procesů. Vyskytnout se mohou i takzvané Grey-Box testy, které jsou kombinací dvou předešlých.

Black-Box testování

Black-Box testy se netýkají vnitřních procesů softwaru. Na software je pohlíženo jako na celek a testujete se z vnějšího prostředí. Jeho principem je zadání dat do testovaného softwaru a čekání na jeho reakci. Vstupní data se zadávají jak validní, tak i taková, která by softwarem neměla být akceptována. Reakce či její výstup musí být správný, tedy reagovat podle předdefinovaných očekávaných výstupů. Kontrola probíhá pomocí uživateli přístupných rozhraní, tabulek a obrazovek, aniž by se věnovala pozornost programovému kódu nebo modulům tvořícím software. To znamená, že pro tuto metodu jsou především důležité funkční znalosti než ty technické. Black-Box testy se používají hlavně během funkčního testování, tj. systémové a akceptační testy.

White-Box testování

White-Box testy se zaměřují na vnitřní procesy jednotlivých komponent. V průběhu těchto testů lze například testovat, zda jsou moduly správně naprogramovány, zda jsou na disk zapsány správné průběžné výsledky, zda byly dodrženy programovací standardy a zda kód není zbytečně komplikovaný nebo dokonce zbytečný. Je také možné testovat, zda jsou správné moduly vyvolány pomocí správných parametrů.

Pro White-Box testy je nutná znalost vnitřní struktury softwaru. White-Box testy se používají hlavně při testech komponent a testech integrace komponent [SCHOTANUS, 2009].

Grey-Box testování

Kombinací Black-Box testu a White-Box testu je právě Grey-Box test. V případě Grey-Box testů musí mít tester jak funkční znalosti, tedy musí vědět, jaké výstupy očekávat na základě zadaných vstupů, tak musí znát i základy vnitřních procesů softwaru a jeho strukturu. Informace o vnitřních procesech jsou jen základní, což znamená, že nelze

hovořit o White-Box testu, ale současně jsou nad úrovní Black-Box testu, proto se používá termín Grey-Box test, který je v praxi vcelku frekventovaný [TESTOVÁNÍ SOFTWARE, 2020].

3.2.3 Manuální a automatizované testování

Podle způsobu realizace testů rozlišujeme dva typy testů, a to manuální a automatizované testování. V souvislosti s manuálními testy rozlišujeme ještě třetí typ testů, kterými jsou takzvané exploratory testy.

Manuální testy

Jedná se o testy, která provádí tester, podle již vytvořených test casů. Tyto testy jsou sice časově náročnější a nemusí být odhaleny všechny defekty, jelikož zde zasahuje z části lidský faktor, ale mohou být naopak užitečné při odhalování defektů, které může odhalit jen lidský úsudek. Jako jsou například grafické defekty. Manuální testy se provádí u testování objektů, které se pravidelně neopakují, zpracovávají menší množství dat nebo je u testu vyžadována právě kontrola člověkem.

Exploratory testy

Exploratory testy jsou taktéž prováděné testerem, ale od těch manuálních se liší způsobem provedení. Netestuje se totiž podle předem napsaných test casů. Jeho záměrem je zkoumat, jak daný software funguje, jaké jsou jeho vlastnosti a jak se chová.

Automatizované testy

Pokud má testovaný software vytvořené stabilní manuální test casy, podle kterých se testuje opakovaně, je otázkou, zda takové testy neautomatizovat. Podstatou automatizovaného testování je to, že se provádí pomocí programu, tedy bez přímého zásahu člověka.

Hlavní výhodou automatizovaných testů je časová úspora, možnost spuštění testu přes noc, a zefektivnění prováděných postupů při testování. Navíc bez zásahu člověka se eliminuje možnost přehlédnutí případného defektu a tím se potencionálně zvyšuje kvalita produktu.

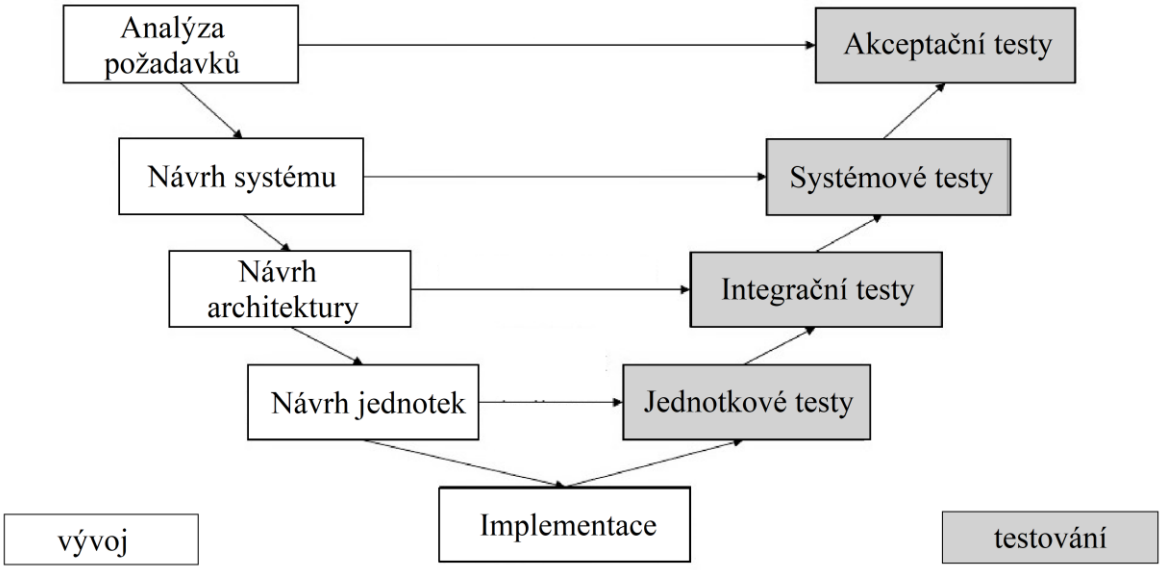
Nevýhodou je nutnost aktualizace testů pokaždé, je-li provedena nějaká změna v objektu, kterého se testy týkají. Když se ihned neaktualizují, vzniká riziko propuštění defektu do provozu. Další nevýhodou může být i obtížná automatizace testů u softwaru, který již existuje. Vhodnější je počítat s automatizovanými testy při vývoji softwaru.

Automatizované testy jsou výhodné pro software, u kterého se předpokládá delší životnost, pro krátkodobé, které mají životnost v řádech měsíců, se takové testy nevyplatí.
Jsou vhodné pro testy, které se často nemění, ale často se spouští. Jedná se například o smoke testy nebo regresní testy [TESTOVÁNÍ SOFTWARU, 2020].

3.3 Typy testů podle úrovně vývoje

V průběhu vývoje softwaru vznikají odlišné požadavky na testování. Jak je vidět na Obrázku 2, jsou rozlišovány čtyři úrovně testování, a sice testování jednotek, integrační testování, systémové testování a akceptační testování, kdy každý test odpovídá jedné z vývojových fází.

Obrázek 2 V – model



Zdroj: Vlastní zpracování dle SOFTWARE TESTING, 2015

3.3.1 Testování jednotek – Unit testy

První úrovní testů je test jednotek neboli unit testy, který se provádí po ověření kódu programátorem. Spočívá v testování nejmenších částí programu, tedy takzvané jednotky. Dle programovacího jazyka, buď procedurální nebo objektově orientovaný lze pod pojmem jednotka chápat funkci, proceduru, metodu nebo třídu.

Podstatou testování jednotek je testování každé jednotky zvlášť, nezávisle na dalších jednotkách. Separace jednotky od ostatních je vyžadována z důvodu zamezení ovlivňování výsledků jinými jednotkami.
Tato úroveň testu se provádí především na začátku vývoje, protože v pozdějších fázích je takové testování špatně aplikovatelné a často vyžaduje refaktoring kódu (zlepšení existujícího kódu beze změny vnějšího chování kódu) či jiné úpravy.
Na této úrovni testování, tedy na počátku vývoje je zpravidla oprava nalezeného defektu nejlevnější, proto je dobré tuto fázi nepodceňovat.

3.3.2 Integrační testování

Po úspěšně otestovaných jednotkách je třeba moduly takových jednotek vzájemně integrovat, aby mohl vzniknout systém jakožto celek. Integrační testování probíhá jako testování interakcí mezi moduly v rámci vzniklého celku. Integrace modulů do jednoho funkčního celku – systému může probíhat čtyřmi způsoby a sice:

- **Velký třesk** – všechny jednotky integrovány najednou.
- **Přístup shora-dolů** – dle hierarchie, nejdříve nejvyšší modul, pak až nižší (systém může zčásti fungovat).
- **Přístup zdola-nahoru** – dle hierarchie, nejdříve nejnižší modul, pak až vyšší (systém funguje až po integraci nejvyššího modulu).
- **Kombinovaný přístup** – dle hierarchie, hlavní moduly – nejdříve nejvyšší, často využívané moduly – nejdříve nejnižší, zbývající moduly na střední úrovni.

Tento typ testování je částečně obsažen ve většině testovacích úrovní, proto se u menších projektů na tuto fázi neklade velký důraz.
Integrační testy se dají provádět i pro integraci systémů, které již prošly systémovými testy. Pak se jedná o testy systémové integrace (system integration tests – SIT).

3.3.3 Systémové testování

Zcela zásadní úroveň testování je systémové testování, které nastává po ověření úspěšné integrace. Na této úrovni je důkladně testováno, zda systém naplňuje všechny zákazníkovy požadavky, tedy zda reálné chování systému naplňuje očekávané chování systému. Vyvíjený software se testuje z pohledu zákazníka, podle předem napsaných testovacích

scénářů. Jedná se o poslední fázi testování před předávkou produktu zákazníkovi, takže lze považovat tyto testy za jakousi výstupní kontrolu softwaru.
Mezi systémové testy se řadí například:

- **Funkční testy** – ověřuje, zda systém splňuje definované požadavky.
- **Testy robustnosti** – ověřuje, jak pracuje s chybnými vstupy či nečekanými situacemi.
- **Testy použitelnosti** – ověřuje uživatelskou přívětivost systému.
- **Výkonnostní testy** – ověřuje výkon systému za pomocí sledování určitých ukazatelů.

3.3.4 Akceptační testování

Poslední úrovní testování, vzhledem k vývoji softwaru, je akceptační testování. Testování probíhá na straně zákazníka a označuje se zkratkou UAT neboli User Acceptance Testing. Zákazník, respektive uživatelé z jeho strany, ověřuje, zda jsou splněna všechna jeho předem stanovená akceptační kritéria, tedy předem domluvené ověřitelné podmínky pro přijetí softwaru. V případě, že nedojde ke splnění akceptačních kritérií, má zákazník možnost softwaru odmítnout.
Testování probíhá na testovacím prostředí u zákazníka, a to za pomocí předem připravených scénářů zahrnující běžné aktivity v rámci provozu. V případě nalezení defektu je podán report vývojovému týmu, který defekty v co nejkratším čase vyhodnotí, opraví a nasadí zpátky na testovací prostředí u zákazníka [ROUDENSKÝ, HAVLÍČKOVÁ, 2013].

3.4 Další používané testy

Dalšími testy, které jsou nedílnou součástí testovacího procesu, jsou testy regresní a konfirmační. Jedná se o testy, které mají za úkol ověřit test case nebo celý testovací scénář, který byl již testován, ale je třeba ho z různých důvodů otestovat znovu. Naopak testy, které ověřují software ještě před hlavním testováním jsou smoke testy.

3.4.1 Smoke testy

Smoke testy se využívají pro ověření, že testovaný software nebo jeho určitá část splňuje základní požadavky. Jejich cílem je zajistit, aby fungovaly nejdůležitější funkce ještě před zahájením hlavní fáze testování. Používají se hlavně před zahájením funkčního testování.

3.4.2 Regresní testy

Významnou složkou testování je právě regresní testování. Jeho úkolem je otestovat již otestované části softwaru, které nebyly modifikovány, ale modifikací jiných částí či implementací nových prvků se u nich mohly znovu objevit již dříve opravené defekty nebo defekty zcela nové. Ty mohou vzniknout například na základě změny v kódu nebo konfiguraci prostředí.
Při regresním testování se zpravidla objevují tři typy defektů:

- **Lokální defekty** – vnikají změnou v kódu u dané komponenty.
- **Vzdálené defekty** – změnou v jedné komponentě vznikne defekt u jiné komponenty.
- **Odkryté defekty** – objeví se defekt, který již dříve existoval, ale projevil se až změnou v softwaru.

Regresní testy se mohou provádět na pěti různých úrovních a sice:

- **Testování jednotek** (Unit testy) – ve většině případů automatizované, protože se předpokládá vysoká četnost spuštění testů.
- **Testování integrace komponent** – ve většině případů automatizované a je spouštěno po tvorbě nového sestavení softwaru či po uložení nové verze kódu do systému správy verzí.
- **Systémové testování** – ve většině případů manuální, a to především při testování produktu s grafickým uživatelským rozhraním (GUI), automatizované jen zřídka.
- **Testování integrace systémů** (SIT) – provádí se jak manuálně, tak i automatizovanými testy. V případě automatizovaných testů je vhodné je spouštět vždy v noci před tím, než začnou testeři testovat manuálně.

- **Uživatelské akceptační testování** (UAT) – má za úkol ověřit, že změny v softwaru souhlasí s definovanými či očekávanými potřebami uživatelů. Dále se ověřuje, zda stěžejní funkce, které jsou neměnné, zůstaly funkční a neobjevil se u nich žádný defekt. Report z dokončených regresních testů, které proběhly úspěšně, je často i jednou z akceptačních podmínek pro akceptaci nové verze softwaru, která byla vyvinuta.

3.4.3 Konfirmační testování

Regresními testy jsou mnohdy zaměňované s konfirmační testy. Ovšem konfirmační testování někdy také nazývané jako retestování je založeno na opětovném testování test casu, v rámci kterého byl identifikován defekt. Má za úkol ověřit, zda se oprava defektu zdařila či nikoliv [BUREŠ, RENDA, DOLEŽEL, aj., 2016].

3.5 Dokumentace v testování

Aby mělo testování určitý řád a nestávalo se jen náhodným klikáním a zkoušením testovaného produktu, je třeba mít vytvořenou testovací dokumentaci. V této dokumentaci by měly být vymezeny všechny elementy spojené s testováním, a to například vymezený cíl testování, technické požadavky, typy používaných testů atd.
Jelikož má tester zodpovědnost za bezchybnost produktu po jeho otestování, ačkoliv není tvůrcem případných defektů, je pro něj vytvořená dokumentace významnou pomůckou pro redukci počtu defektů, které se testováním nezachytí. Také slouží jako výstupní protokol o provedených testech a o podmínkách za kterých byly vykonány, čímž umožní daný testovací scénář zopakovat a tím provést test za totožných podmínek. Tvorba takové dokumentace by tedy měla být běžnou součástí projektového plánu při vývoji produktu.
Testovací dokumentaci představují například testovací plány, testovací případy, testovací scénáře nebo samotné reporty defektů [TESTOVÁNÍ SOFTWARE, 2020].

3.5.1 Testovací plán

Důležitým dokumentem, který by měl být vytvořen ještě před začátkem testování softwaru, je testovací plán. Jsou v něm uvedeny všechny základní informace o procesu a podmínkách testování konkrétního softwaru. Testovací plán by měl obvykle obsahovat těchto šest oblastí:

- **Cíl testování** – definování toho, co se od testování softwaru očekává nebo čeho má být dosaženo. Cíle mohou být různé, od otestování nově přidané funkce po kompletní otestování celého softwaru.
- **Přehled plánovaných oblastí k testování** – vychází z testovacích požadavků. Například z use case, což je příklad užití softwaru, nebo z komunikace se zákazníkem. Snaží se utvořit takový soubor testů, který zahrne všechny funkčnosti aplikace s přihlédnutím k použití ze strany zákazníka.
- **Stanovení priorit** – Priority se stanovují pro jednotlivé prvky testovaného softwaru i pro jednotlivé testy. Nejvyšší prioritu dostávají takové prvky, které jsou nejvýznamnější pro fungování softwaru, anebo ty prvky, jejichž nefunkčnost může být největším rizikem při dosahování cílů. Testování nadále probíhá sestupně, dle priorit.
- **Testovací strategie** – obsahují popis typů testů, které mají být provedeny, techniky, kterými mají být provedeny a způsob jejich vyhodnocení. Testy jsou rozřazeny do jednotlivých úrovní testování, dle fáze vývoje.
- **Požadavky na zdroje** – jedná se o identifikaci požadavků, které je potřeba splnit, aby bylo možné provést testy. Spadají sem jak požadavky technické, tak i požadavky na lidské zdroje, tedy požadavky na složení testovacího týmu.
- **Definice rizik** – soupis situací, které mohou potencionálně ohrozit úspěšné testování, míra jejich významnosti a opatření proti jejich vzniku. Mezi takové hrozby lze zařadit například nedodržení termínu dodání softwaru či nedostatek pracovníků [TESTOVÁNÍ SOFTWARE, 2020].

3.5.2 Test Case

Test case neboli v překladu testovací případ je dokument nebo záznam s popisem toho, jak provést vybraný test. Jedná se o soubor, kde jsou popsány konkrétní kroky, jak postupovat při testování určité softwarové komponenty. V případě manuálního testování obsahují test casy soubor jednotlivých kroků, které zahrnují popis postupu testování a jeho očekávané výsledky. U automatizovaných testů jsou test casy představovány sadou programových instrukcí a výsledek, zda byl test úspěšný nebo zda byl objeven defekt, by měl vyhodnotit sám.

Test case má své náležitosti, které by měl obsahovat. Správný test case by měl určitě mít jasný název, ze kterého bude jasné, co má příslušný test case otestovat, aniž by bylo potřeba projít jednotlivé kroky. Musí mít své unikátní ID, prioritu testu a případně předpoklady pro uskutečnění testu. Dále obsahuje jednotlivé kroky, s popisem činnosti a popisem toho, jak by měl vypadat očekávaný výsledek. Po provedení každého kroku je nutné zapsat, jaký byl aktuální výsledek a jaký je výsledek testu, tedy jestli prošel či byl objeven nějaký defekt.

Tabulka 1 Příklad šablony pro vytvoření test casu

TEST CASE					
Název			**ID**		
Popis					
Prerekvizity			**Priorita**		
Číslo kroku	**Činnost**	**Vstupy**	**Očekávané výstupy**	**Výsledek testu**	**Komentář**

Zdroj: Vlastní zpracování dle SOFTWARE TESTING HELP, 2019

Je velice důležité vytvořit kvalitní test case, protože každá chyba při psaní test casu jej může zároveň znehodnotit. Přitom tvorba kvalitních test casů není zcela snadná. Mezi nejčastější nedostatky při psaní test case patří:

- **Chybějící kroky** – vynechání kroku, který se zdá být samozřejmý.
- **Příliš podrobný popis** – zbytečně dlouhé a podrobné popisování každého kroku způsobuje horší čitelnost a orientaci v testu.
- **Nadměrné užívání žargonu** – používání zkratek a termínů, které nejsou známy těm, co provádí testy.
- **Nejasná kritéria pro vyhodnocení výsledků** – nedostatečný popis očekávaného výstupu testu nebo konkrétního kroku [PAGE, JOHNSTON, ROLLISON, 2009].

3.5.3 Testovací scénář

Testovací scénář se sestává z podrobného testovacího postupu. Je tvořen souborem test casů, které jsou na sobě nezávislé a je možné je zařadit v testovacím scénáři libovolně nebo na sebe navazují, což znamená, že musí být seřazeny v určitém pořadí a výstupy z jednoho

test casu jsou nutným vstupem pro test case navazující. Úkolem testovacích scénářů je simulovat určité způsoby užívání testovaného produktu, které by měly pokrýt otestování všech požadavků.

V případě, že není na psaní test case čas, lze pracovat pouze s podrobnými testovacími scénáři, které jsou důležitější než test casy [SOFTWARE TESTING CLASS, 2014].

Příklad toho, jak může vypadat sepsaný testovací scénář, je v následující tabulce, kde jsou pro ilustraci uvedené pouze dva test casy, ale v praxi mohou být až desítky test casů v jednom testovacím scénáři.

Tabulka 2 Příklad tvorby testovacího scénáře

ID Testovacího scénáře	Testovací scénář	ID Test casu	Test case	Kroky	Testovací data	Očekávaný výstup	Aktuální výstup	Výsledek
TS01	Kontrola přihlášení uživatele	**TC01**	Kontrola přihlášení uživatele s validními daty	1. Jdi na stránku www.prihlaseni.cz 2. Zadejte uživatelské ID 3. Zadejte heslo 4. Klikněte na "Potvrdit"	Uživatelské ID = demo_user 1 Heslo = pass18	Uživatel by se měl být schopný přihlásit do aplikace	Podle očekávání	Prošlo
		TC02	Kontrola přihlášení uživatele s nevalidními daty	1. Jdi na stránku www.prihlaseni.cz 2. Zadejte uživatelské ID 3. Zadejte heslo 4. Klikněte na "Potvrdit"	Uživatelské ID = demo_user 1 Heslo = Glass18	Uživatel by NEMĚL být schopný přihlásit do aplikace	Podle očekávání	Prošlo

Zdroj: Vlastní zpracování dle REQTEST, 2018

3.5.4 Report defektu

Velice důležitým dokumentem při testování je report defektů. Report by měl být srozumitelný, konkrétní a přesný tak, aby byl defekt snadno reprodukovatelný. V opačném případě může vyjasnění defektu mezi testerem a vývojářem vyžadovat rozsáhlou komunikaci nebo několik hodin práce navíc.

Položky, které by měl mít kvalitní report o defektu:

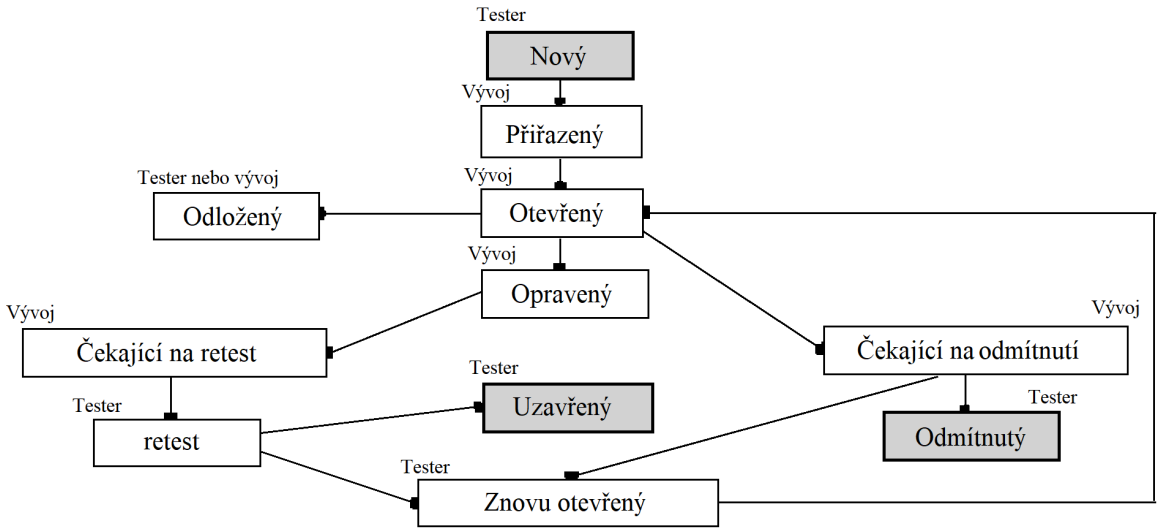
- **Název** – nejdůležitější položkou při testování je právě jeho správně napsaný název. Význam má především při hledání vytvořeného reportu nebo při hledání oblasti nebo typu chyb, které se vyskytují u vyvíjeného produktu. Měl by proto obsahovat přesné shrnutí identifikovaného defektu.

- **Popis** – souhrn informací o defektu, které se nevešly do názvu reportu. Například jak by se měl produkt dle očekávání chovat a jaké je jeho skutečné chování.
- **Stav** – popisuje stav, ve kterém se řešení defektu nachází. Zda je otevřený, aktivní, uzavřený atd.
- **Číslo chyby** – číslo verze testovaného produktu, ve kterém byl defekt nalezen. Je užitečný například pro reprodukci defektu nebo ověření oprav.
- **Oblast funkcionality** – oblast či podoblast produktu, ve které se defekt nachází.
- **Přiřazení** – přiřazení osoby, která bude zodpovědná za řešení defektu k příslušnému reportu. Přiřazena je vždy jen jedna zodpovědná osoba.
- **Závažnost** – vyjadřuje míru dopadu defektu na zákazníka a na vývoj. Zohledňují se jak důsledky problému, tak i frekvence výskytu defektu a jeho reprodukovatelnost. Většinou jsou ohodnoceny na stupnici od jedné do čtyř, kdy číslo jedna jsou nejzávažnější chyby, a naopak číslo čtyři ty nejméně závažné.
- **Dopad na zákazníka** – popis toho, jak se defekt projevuje z hlediska koncového uživatele.
- **Prostředí** – podmínky, za kterých byl nalezen defekt. Specifikace použitých nástrojů a procesů, verze systému, typ hardwaru atd.
- **Kroky vedoucí k reprodukci** – stručný popis kroků, díky nimž je možné defekt nasimulovat. Měly by být sepsány tak, aby byl kdokoli, kdo se defektem zabývá, schopný podle sepsaných kroků defekt reprodukovat.
- **Výsledek** – vyplňuje se až když je chyba vyřešena. Je zde popsán výsledek defektu. Zda byl opraven, je duplicitní, nelze reprodukovat, byl odložen nebo je popsané chování produktu záměrné [PAGE, JOHNSTON, ROLLISON, 2009].

Report defektu, který má již všechny náležitosti, označí tester jako „nový“. To znamená, že už je zanesen v systému a je na vývojovém oddělení, aby daný defekt přiřadili konkrétnímu vývojáři, který se bude defektem zabývat. Když má defekt již konkrétního řešitele, je report označen jako „otevřený“. V této fázi je na vývojáři, aby rozhodnul, jak s defektem naložit. Defekt může být buď odložen k pozdějšímu řešení, přijat a opraven nebo odmítnut z důvodu, že například popisovaný defekt není nežádoucím chováním nebo nebylo možné defekt nasimulovat. V případě, že je defekt odmítnut vývojářem, vrací se testerovi a ten podle situace report defektu označí jako „odmítnutý“, čímž je defekt

uzavřený, nebo ho znovu otevře. Když je defekt přijat a opraven, je připravený na retest, tedy na opětovné otestování. Tester ho pak otestuje a když je defekt již opraven, označí report defektu jako „uzavřený“. Jestliže nastane situace, kdy defekt stále přetrvává, musí tester report defektu znovu otevřít a tak se report opět dostane k vývojáři ve stavu „otevřený“, jak je zřejmé z Obrázku 3.

Obrázek 3 Příklad životního cyklu defektu



Zdroj: Vlastní zpracování dle PRABHU, 2015

Report defektu je tedy vždy označen určitým stavem, ve kterém se aktuálně nachází, což je důležité proto, aby bylo dohledatelné, v jakém stavu se konkrétní defekt nachází nebo kolik defektů je třeba ve stavu „nový“. Přechody mezi jednotlivými stavy mají svá pravidla, která jsou přednastavena v nástroji na správu defektů. To znamená, že nelze dát report defektu do stavu „nový“ ze stavu „odmítnutý“.

Životní cyklus není striktně daný a každá firma si ho může modifikovat dle svých potřeb. Jakmile se ale na jeho podobě všechny zainteresované strany shodnou, musí být po celou dobu trvání projektu neměnný. Pro správu reportů o defektech se využívají speciální softwarové aplikace, kam se reporty píšou a kde jsou uloženy [BUREŠ, RENDA, DOLEŽEL, aj., 2016].

3.6 Testovací tým

Jelikož je testování z části subjektivní záležitost a hraje zde roli lidský faktor, nelze testování provádět pouze jedním člověkem. Za účelem testování jsou sestavovány různě velké týmy. Každý člen týmu má svou specifickou roli. Takových rolí lze identifikovat několik, avšak základem by vždy měl být alespoň test exekutor, test analytik a test manager.

Test exekutor

Náplní práce test exekutora je provádění testů podle test casů. Pořadí, v jakém jsou testy prováděny, je definované na základě návrhu test analytiků nebo na základě jimi vytvořených harmonogramů. Hlavním úkolem test exekutora je hledání abnormalit a rozporů reálného chování objektu s test casy, tedy potencionálních defektů. V případě nalezení takového defektu musí chybu dokumentovat a informovat o ní příslušnou osobu a po opravě defektu příslušný objekt přetestovat.

Test analytik

Test analytik má na starosti návrh a tvorbu test casů, z nichž tvoří testovací scénáře, na základě analýzy podkladů se specifikacemi, které má k dispozici. Je na něm, aby obsáhl vytvořenými testy všechny požadavky zákazníka. Po vytvoření testů přiřazuje test analytik vytvořeným testům prioritu, ve které mají být jednotlivé test casy a scénáře provedeny. Tvoří také sady regresních testů a provádí jejich údržbu. Dokumentuje výsledky provedených testů a provádí analýzy testů. Reportuje na základě výsledku provedených testů dosaženou kvalitu produktu.

Test manager

Osobou zodpovědnou za celý testovací tým je test manager. Vede tým, je zodpovědný za rozpočet, organizaci a všechny aktivity spojené s testováním. To znamená tvorbu testovacích plánů, podávání pravidelných reportů o provedených testech, kvalitě testovaných objektů či případném pokroku. Důležitou součástí je i komunikace se zbytkem testovacího týmu a s projektovým týmem [SCHOTANUS, 2009].

3.6.1 Motivace testerů

Pro zajištění kvalitně odvedené práce je třeba testery motivovat. Testeři mohou být někdy demotivování tím, že není podporováno jejich vzdělání, mají malou šanci na kariérní postup, dále špatným přístupem organizace k řízení kvality obecně nebo nedostatečnou pozorností věnovanou testování.
Jako prostředek motivace ke kvalitně odvedené práci testerů, tedy především těch, kteří provádí manuální testy, slouží například:

- Uznání za dobře odvedenou práci.
- Zájem o názory testerů, zapojení do řešení problémů, které mají souvislost s náplní jejich práce.
- Pevná pravidla, která jsou jasně vymezená.
- Diskuse s testery o jejich spokojenosti, čímž dostanou nadřízení zpětnou vazbu a případné podněty ke zlepšení pracovních procesů či spokojenosti testerů.
- Rozmanitost práce, nové pracovní výzvy, zamezení monotónnosti práce.
- Možnost vzdělávání se a zdokonalování a s tím spojený možný kariérní postup.
- Neformální soutěže. Například o nejzajímavější nalezený defekt nebo o nejlepšího testera, což by bylo vyhodnoceno podle předem stanovených kritérií. Tento způsob motivace je vcelku nezvyklý, ale dle praxe se ukazuje, že je dosti účinný, a to především ve větších skupinách testerů.

Naopak motivátor, který není zcela vhodný, je finanční odměna. Například odměna za každý nalezený defekt, který má střední a vyšší závažnost. Tester se pak soustředí především na finanční odměnu, tím se dostává do stresu, což kvalitu jeho testování výrazně nezvýší, naopak ji může ještě snížit [ROUDENSKÝ, HAVLÍČKOVÁ, 2013].

3.7 Vývoj softwaru – model Waterfall

Jedním z přístupů k řízení projektu je metodika Waterfall, kterou v roce 1970 definoval Winston. W. Royce. Jedná se o lineární přístup k řízení projektů. Na počátku projektu jsou shromážděny požadavky všech zainteresovaných stran a zákazníků a následně je vytvořen sekvenční plán projektu, vyhovující těmto požadavkům. Charakteristickým prvkem této metodiky je právě sekvenční přístup k jednotlivým fázím projektu. [VODOPÁDOVÝ

MODEL, 2020] To znamená, že do následující fáze projektu lze vstoupit až tehdy, kdy je současná fáze dokončena a uzavřena.
Metodika Waterfall rozlišuje sedm základních fází a sice:

- **Specifikace požadavků**

Při vývoji nového softwaru je důležité si nejprve nastínit požadavky na konečný produkt. Všechny požadavky jsou shromážděny na začátku projektu, díky čemuž lze naplánovat všechny následující fáze bez dalšího zapojení zákazníka, a to do té doby, než je produkt kompletní. Ovšem jen za předpokladu, že lze všechny požadavky ve fázi specifikaci požadavků shromáždit.

- **Návrh**

Tato fáze zahrnuje návrh základní architektury, která bude následně použita v dalších fázích. V případě, že nebyly v předchozí fázi požadavky řádně identifikovány a promyšleny, nelze ve fázi návrhu software správně vytvořit jeho architekturu.

- **Implementace**

V této fázi už se začíná tvořit funkční software. S vytvořeným designem produktu z předchozí fáze lze začít s kódováním, tedy tvorbou softwaru. Výstupem této fáze by pak měl být funkční software, který dělá to, co bylo dohodnuto v požadavcích. S tím, že se předpokládá, že nově vytvořený software není zcela bez chyby.

- **Integrace**

Po vytvoření funkčního softwaru je potřeba ho integrovat do systému, se kterým bude používán. Integrace je důležitým krokem, protože díky ní je možné vidět, jak bude nový software spolupracovat s jiným, již existujícím softwarem, protože značná část nově vytvořených softwarů je navržena tak, aby spolupracovala s dalším softwarem a byla tak součástí většího celku. Tato integrace slouží spíše pro účely testování, není trvalá.

- **Testování**

Po integraci softwaru do systému je třeba začít fázi testování, protože téměř nikdy není nově vytvořený software bezchybný. Je nutné software otestovat, identifikovat případné defekty a následně je opravit. Testování je velice důležitou fází, která se často podceňuje,

ale je velice důležitá, protože když se defekt neidentifikuje hned, tak nákladovost na jeho opravu stoupá, a to jak finanční, tak časová.

- **Instalace**

Po dokončeném otestování je software připraven k instalaci. V případě dobře navrženého softwaru je přechod na nový software relativně rychlý. V této fázi je třeba proškolit jeho uživatele, aby byl software správě využíván.

- **Údržba**

Poslední fází je údržba softwaru. I když už je software v provozu, stále je třeba plnit požadavky na údržbu, které se objeví v průběhu jeho používání. Software je produkt, který je potřeba neustále monitorovat a zlepšovat tak, aby byl pro zákazníka stále přínosný. Je to z důvodu neustálého vývoje, ať už v rámci vnitřních procesů na straně zákazníka nebo vývoje vnějšího prostředí, které je nutné zohlednit, jinak se stává software postupně zátěží. [THE WATERFALL MODEL, 2020]

Tento model může být úspěšný pouze tehdy, je-li počátečním fázím věnována dostatečná pozornost a je-li na konci každé fáze zajištěná maximální možná kompletnost a validace příslušné fáze.

Za nevýhodu Waterfall modelu lze považovat skutečnost, že u rozsáhlejších projektů téměř nelze dokončit jednu fázi a pokračovat k další, aniž by bylo v budoucnu potřeba se vrátit k té předchozí. Další nevýhodu je téměř nulová možnost reagovat na změny požadavků ze strany zákazníka v průběhu vývoje, protože je software předán zákazníkovi až ve fázi, kdy je nemožné se vrátit do fáze úprav nebo by to bylo velice nákladné. Za nešťastné je považováno i testování, které probíhá až po dokončené implementaci, kdy je software už hotový, protože oprava nalezených defektů je náročnější než nalezení defektu v předchozích fázích. Na druhou stranu je model Waterfall jednoduchý a přehledný. Výhodným se stává především u menších projektů, kde se nepředpokládají následné změny funkcí či vlastností softwaru.

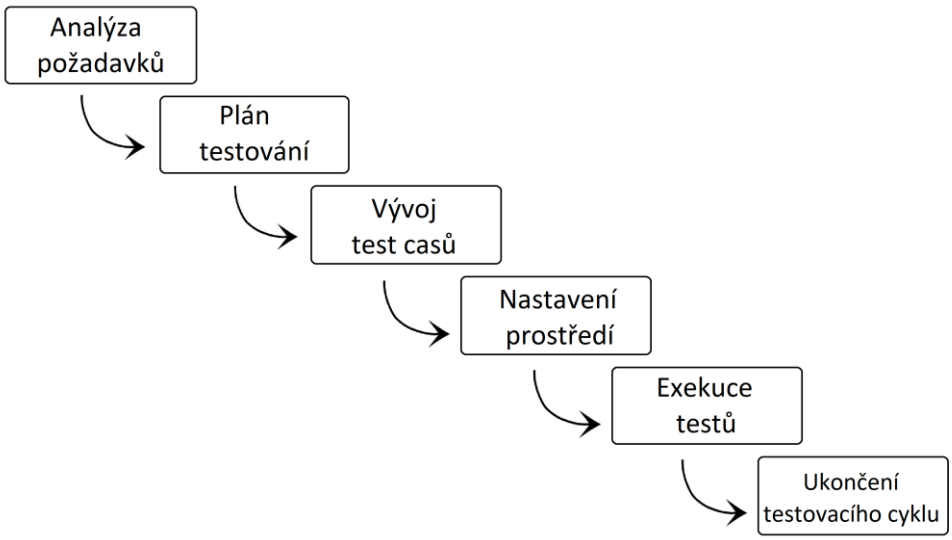
V současné době se model Waterfall při vývoji softwaru téměř nepoužívá, ale využívají se jeho různé modifikace [TESTOVÁNÍ SOFTWARU, 2020].

3.8 Životní cyklus testování softwaru

Životní cyklus testování softwaru, známý pod zkratkou STLC vyplývající z anglického překladu Software Testing Life Cycle, je procesem testování, který se provádí systematickým a plánovaným způsobem, skládá se z řady fází, které je třeba provést v určitém pořadí, protože na sebe téměř všechny fáze navazují svými výstupy. Životní cyklus není striktně daný, v praxi mohou být některé fáze odlišné. Vždy ale zůstává stejný základ.

Na následujícím obrázku jsou jednotlivé fáze životního cyklu testování softwaru.

Obrázek 4 Životní cyklus testování softwaru



Zdroj: Vlastní zpracování dle SOFTWARE TESTING CLASS, 2013

- **Analýza požadavků**

První fází životního cyklu je analýza požadavků. U této fáze je důležité, aby tým pro zajištění kvality, takzvaný QA tým (Quality Assurance), zjistil požadavky a co nejlépe jim porozuměl, aby bylo jasné, co je třeba testovat. Jestliže by došlo ke konfliktu, nesprávnému porozumění požadavku nebo by některý požadavek zcela chyběl, musel by se QA tým snažit vyřešit daný problém s vybranými zúčastněnými stranami, jako je například technický manažer, systémový architekt, business analytik nebo klient, aby lépe porozuměl požadavku. Požadavky mohou být funkční i nefunkční, což je třeba požadavek na výkon či bezpečnost.

- **Plán testování**

Nejdůležitější fází životního cyklu je plánování testování, kde je definována kompletní strategie testování. Proto je tato fáze někdy nazývaná jako fáze strategie testování. Tvorbu plánu testování má obvykle na starosti manažer testování. Podílí se například na odhadu úsilí a nákladů na testování pro celý projekt, na soupisu cíle testování a tvorbě přehledu plánovaných testů. Tuto fázi lze zahájit až po dokončení sběru a analýzy požadavků. Výsledkem plánu testovaní je dokument v podobě test plánu nebo testovací strategie.

- **Vývoj test casů**

Vývoj test casů je zahájen po dokončení fáze plánování testování. Testovací tým vytvoří podrobné test casy a případně i testovací data, pokud jsou pro testování požadována. Jakmile jsou test casy připraveny, je možné je zkontrolovat a posoudit jejich relevanci, což zajišťují kolegové nebo vedení.

- **Nastavení prostředí**

Další velice důležitou fází životního cyklu testování softwaru je nastavení testovacího prostředí. To rozhoduje, za jakých podmínek se bude software testovat. Do procesu nastavování testovacího prostředí se testovací tým příliš nezapojuje, proto je možné tuto fázi zahájit souběžně s vývojem test casů. Nastavení testovacího prostředí zajišťují ve většině případů vývojáři, ale může být zapojen i zákazník. Úkolem testovacího týmu je připravit smoke test casy, díky kterým se ověří připravenost nastaveného testovacího prostředí.

- **Exekuce testů**

Po úspěšném dokončení přípravy vývoje test casů a nastavení testovacího prostředí je možné zahájit fázi exekuce testů. V této fázi lze zahájit samotné testování, tedy spouštění připravených test casů na základě připraveného plánu testování. Provádění testů mají zpravidla na starosti test exekutoři nebo test analytici.

Po provedení testu je důležité ho označit příslušným stavy, jejichž názvy a významy jsou předem dohodnuty. Tedy v případě, že při provádění testu odpovídalo chování testovaného softwaru danému test casu, může být označen jako úspěšný. Jako neúspěšný lze označit test case, při jehož testování byl nalezen nějaký defekt, tedy když chování testovaného softwaru se neshoduje s popisem v test casu. Takový defekt musí být nahlášen vývojovému

týmu a může být i spojen s příslušným test casem pro další analýzu. K nahlašování chyb se využívají různé systémy pro sledování defektů. Jestliže je některý z test casů blokován v důsledku nějaké vady, označí se test case jako blokovaný.

Tímto způsobem lze získat přehled o úspěšnosti spuštěných test casů a tím i o kvalitě testovaného objektu. Po odstranění defektů lze tytéž test casy spustit znova a ověřit, zda oprava defektů proběhla úspěšně.

- **Ukončení testovacího cyklu**

V poslední fázi životního cyklu testování by se měli setkat členové testovacího týmu a vyhodnotit kritéria pro dokončení testovacího cyklu. Hodnotit lze například podle pokrytí testů, kvality, nákladů a času.

Měla by proběhnout i diskuse o tom, co šlo dobře a kde je naopak prostor pro vylepšení, což může sloužit jako vstup do nadcházejících testovacích cyklů a může pomoct posílit slabá místa v procesu testovacího cyklu. Po dokončení testovacího cyklu je potřeba připravit zprávu o uzavření testovacího cyklu [SOFTWARE TESTING CLASS, 2013].

4 Vlastní práce

Vlastní práce je zaměřena na popis současného stavu testování. Dále na identifikaci problémů a slabých míst, které se vyskytují při uživatelském testování, a následně je popsán návrh na změny, jež by mohly zlepšit současný stav a vyřešit identifikované problémy.

4.1 Charakteristika společnosti

Sazka a.s. (dále jen Sazka) je loterijní společnost s dlouholetou tradicí. Jedná se o jedničku na trhu v oblasti sázení. Byla založena roku 1956, kdy uvedla na trh svou první hru. Jmenovala se stejně jako společnost a spočívala v tipování výsledků sportovních utkání. Následující rok začala firma fungovat jako účelové zařízení Československého svazu tělesné výchovy a sportu [SAZKA, 2019].

V současné době nabízí Sazka celou řadu produktů – loterie, losy, kurzové sázky a vcelku nově také technické hry. Ty se řadí do hazardních produktů. Všechny tyto možnosti jsou k dispozici na webových stránkách. Loterie a losy jsou přístupné i v mobilní aplikaci Sazka. Kurzové sázky mají svojí vlastní aplikaci a sice aplikaci Sazkabet. Kromě Sazka her a kurzových sázek jsou všechny zmiňované možnosti dostupné na terminálech Sazky, které jsou rozmístěné po celé České republice. Ke službám a produktům, které jsou nehazardní, patří služby SAZKAmobil, dobíjení kreditů či prodej vstupenek.

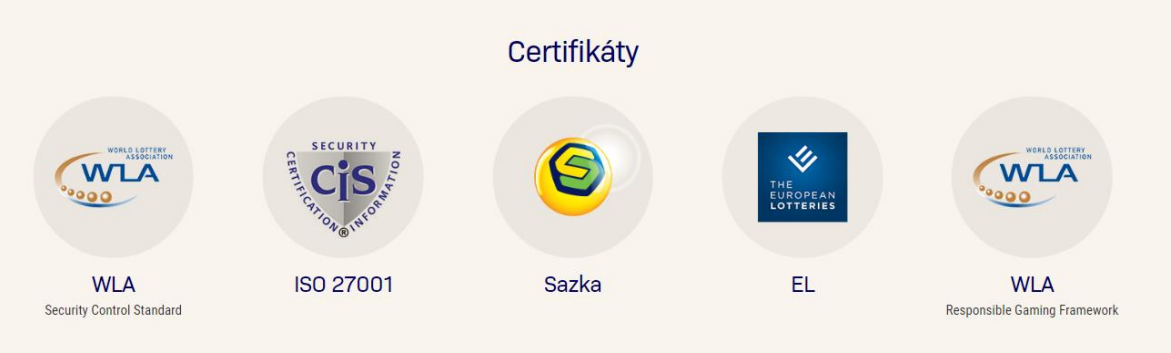
Mimo sázek a dalších služeb jsou zájmy Sazky velice široké a společensky velmi záslužné. Vede například projekt Sazka olympijský víceboj, který má za úkol vést děti základních škol ke sportu a k aktivitám s ním spojeným. Podporuje i sociálně znevýhodněné děti prostřednictvím projektu Česká olympijská nadace. Olympijskou podporu však Sazka započala mnohem dřív, a to konkrétně v roce 1981, kdy byl výtěžek ze hry Sázka 5 ze 40 věnován na přípravu a pobyt československých olympijských reprezentantů na zimní a letní olympiádu. Tuto podporu olympioniků udržuje dodnes [SAZKA,2020].

I když se jedná o společnost, jejímž hlavním podnikatelskou činností jsou loterie, kurzové sázky či technické hry, tedy hazardní činnosti, snaží se být Sazka zodpovědnou společností. Umožňuje na základě zákona o hazardních hrách nastavit hráčům herní limity. Jedná se o finanční limity, limity délky hraní a limity počtu přihlášení. Po překročení

nastavených limitů není umožněno pokračovat v aktivitách, které se limitů týkají, což přispívá k eliminaci rizikových faktorů hazardních her.

Skutečnost, že je Sazka zodpovědnou společností, dokládá fakt, že získala řadu certifikátů, které jsou zobrazeny na Obrázku 5. Má například mezinárodní certifikát zodpovědného hraní – European Lotteries (EL) a certifikát zodpovědného hraní nejvyššího stupně – World Lottery Association (WLA).

Obrázek 5 Certifikáty získané společností Sazka a.s.



Zdroj: SAZKA, 2020

I když Sazka pravidelně iniciuje různé sportovní akce pro děti, nejsou děti její cílovou skupinou. Jako cílovou skupinu lze označit všechny dospělé osoby, tedy osoby starší 18 let. V případě, že si chce zákazník založit u Sazky účet, musí splňovat podmínku, že je právě starší 18 let [SAZKA, 2020].

4.1.1 Organizační struktura společnosti

Sazka a.s. v současné době zaměstnává okolo 400 stálých zaměstnanců. Tato společnost je členem společnosti SAZKA Group a.s. Ovládání Sazka a.s. probíhá skrze SAZKA Group a.s. nepřímo. Společnost Sazka a.s. je totiž ovládána mateřskou společností SAZKA Czech a.s.

Orgány Sazka a.s. zahrnují představenstvo, dozorčí radu a valnou hromadu. Členem představenstva je předseda, místopředseda a členové představenstva. V dozorčí radě je předseda dozorčí rady a dva členové dozorčí rady. Třetím orgánem je valná hromada. Valnou hromadu tvoří všichni akcionáři dané společnosti. Tedy v případě Sazka a.s. se jedná o jediného akcionáře, a sice SAZKA Czech a.s. [SAZKA A.S., 2019].

V čele Sazka a.s. je generální ředitel Robert Chvátal, který má pod sebou deset divizních ředitelů. Jedná se konkrétně o tyto divize:

- iGaming – rozvoj online produktů Sazka a.s.
- právní a regulatorní – právní, smluvní a regulatorní agenda společnosti
- prodej – plnění prodejního plánu a řízení a rozvoj prodejní sítě
- marketing – marketingové řízení značky, kampaně a komunikace
- korporátní komunikace – externí a interní komunikace, CSR
- finance – veškeré finanční operace (daně, účetnictví, controlling...), řízení rizik, správa budov a nákupu
- technologie – sledování nejnovějších trendů
- HR – získávání a udržení zaměstnanců
- hry a provoz – provozování a realizace loterijních i neloterijních produktů, losování, kontaktní centrum, tiketing a logistika
- SAZKAmobil a FTS – vedení virtuálního mobilního operátora SAZKAmobil a finanční a transakční služby [SAZKA, 2020]

4.2 Vývoj softwaru a testování

Rozsáhlé portfolio činností Sazky vyžaduje řízení celé řady ICT projektů, neboť s postupující digitalizací je většina spojena s ICT technologiemi. Jde především o vývoj nových her, webu nebo aplikací, bezhotovostní platební styk a další.

Vývoj softwaru probíhá pomocí modifikovaného modelu Waterfall, který má některé fáze vývoje upravené, ale přesto zachovává myšlenku využití sekvenčního přístupu k jednotlivým fázím vývoje. To znamená, že každá fáze musí být zcela dokončena a uzavřena před vstupem do další fáze. Není ale deklarované, že se k předchozí fázi už nelze vrátit. Naopak je vcelku běžné, že právě ve fázi testování je nalezen defekt a je nutné ho vrátit k programátorům, kteří mají na starosti implementaci a kódování. Vývoj probíhá dvěma způsoby. Buď si celý vývoj zajišťuje Sazka sama, což znamená, že se všechny procesy a komunikace odehrávají pouze v rámci firmy nebo je do vývoje zapojena i externí firma, která má na starosti samotný vývoj softwaru ve smyslu vytvoření produktu podle předem definovaných požadavků.

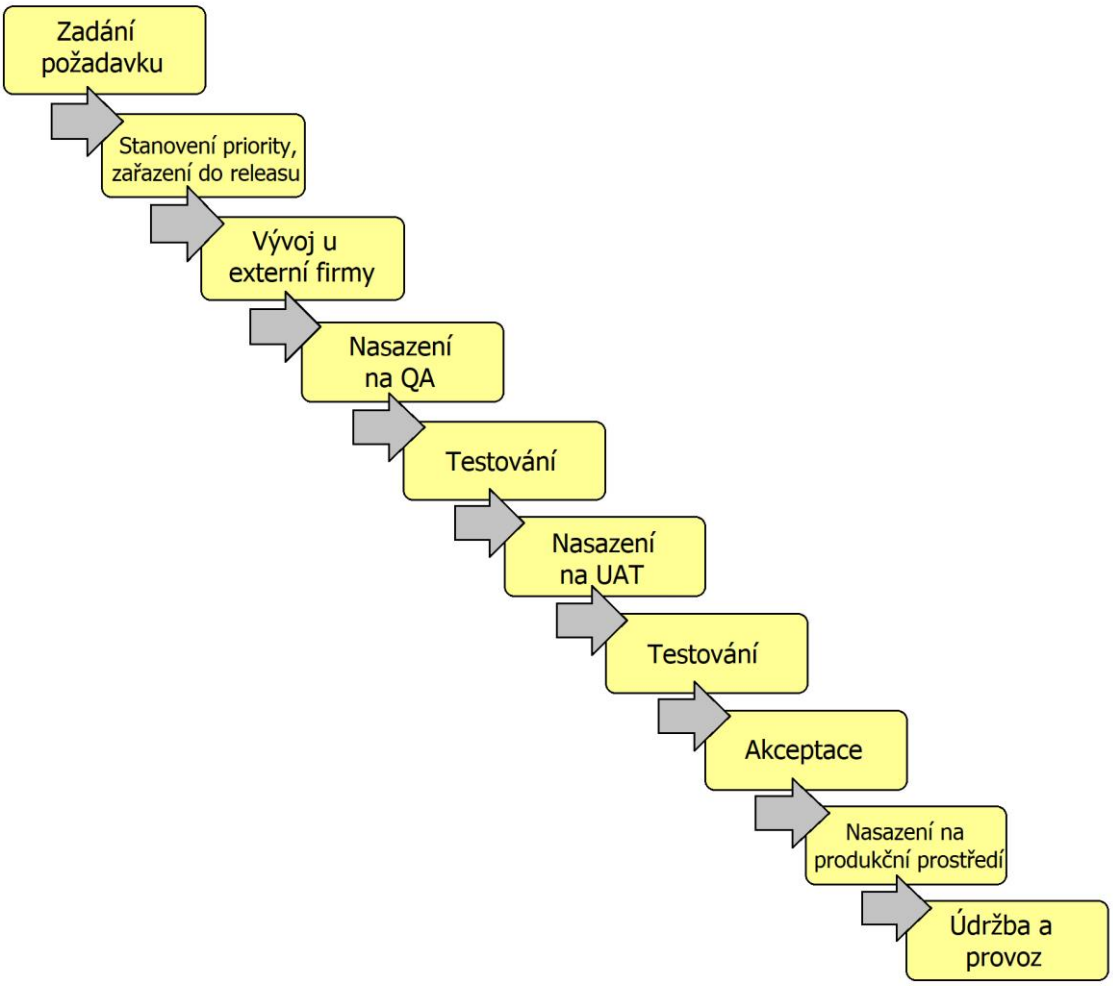
4.2.1 Vývoj her

Vývoj technických her, loterií a losů neboli her, které jsou součástí nabídky na webové stránce, probíhá externě. Dodávku her má na starosti externí firma, jež zajišťuje hry i platformu pro správu uživatelských účtů, které si může běžný uživatel u Sazky založit. Externí firma dodává hotové hry, které vyvíjí třetí strana. Stejným způsobem je vyvíjen i obsah terminálů.

Na testerech v Sazce je pak akceptační testování her. Vývoj probíhá modifikovanou metodikou Waterfall, jak je zřejmé z Obrázku 6, kde jsou popsány jeho jednotlivé fáze. Vývoj je popsán pouze z pohledu Sazky, což znamená, že zde není uvedena fáze návrhu struktury a implementace.

Prvním krokem při vývoji je zadání požadavků Sazky na vyvíjenou hru externí firmě. Zde je vždy nutné zadat požadavky co nejpřesněji, protože externí firma v průběhu vývoje se Sazkou aktuální stav nekonzultuje a Sazka dostává až hotový produkt, tedy není prostor na změny po zadání požadavků. Dále se stanoví priorita zařazení hry do releasu, v jakém by se daná hra měla objevit. Do releasu se dává více her najednou, takže se pak testuje celý balík her, které jsou v příslušném releasu, současně. Po vyjasnění těchto podmínek nastává vývoj na straně externí firmy. Když je hra hotová, je možné ji nasadit na QA (Quality assurance), kde následně probíhá testování. V této fázi se provádí unit testy, systémové tesy a systémově integrační testy. V případě, že je hra na QA úspěšně otestována, nasadí externí firma v rámci stanoveného releasu příslušnou hru na UAT (User Acceptance Testing). Zde se provádí systémové integrační testy (SIT) a uživatelské akceptační testy (UAT). Když je hra na UATu úspěšně otestována, přichází na řadu akceptace vyvinuté hry ze strany Sazky a lze ji nasadit na produkční prostředí, tedy na prostředí, kde je hra dostupná veřejnosti. Posledním krokem je údržba a provoz. Po nasazení na produkční prostředí nelze hru nechat již bez povšimnutí, je třeba ji průběžně monitorovat, udržovat a provádět kontrolní testy – tedy regresní testy.

Obrázek 6 Vývoj her z pohledu Sazky



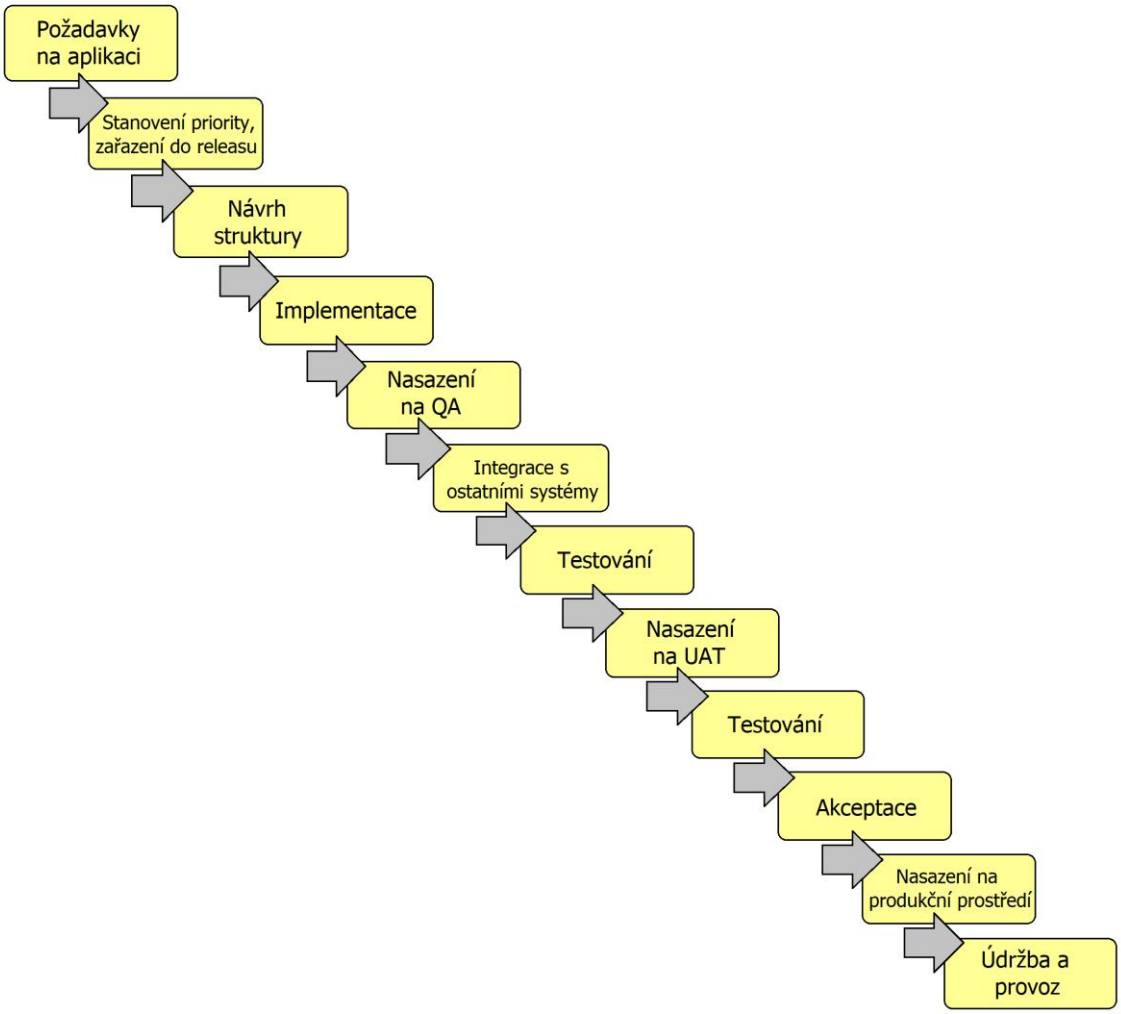
Zdroj: Vlastní zpracování

4.2.2 Vývoj aplikací

Aplikace nebo její části jsou vyvíjeny v rámci Sazky, stejně tak jako webové stránky. Vývoj probíhá také pomocí modifikované metody Waterfall, jak je zřejmé z Obrázku 7, a je z velké části stejný, jako vývoj her. Nejprve jsou sepsány požadavky na vyvíjenou aplikaci, tedy na funkce, design, integrace na jiné systémy atd. Poté se stejně jako při vývoji her stanoví priorita zařazení hry do releasu, v jakém by se daná aplikace měla objevit. Následně je navržena základní struktura vyvíjené aplikace a je možné začít s implementací, tedy se samotnou tvorbou aplikace. Když je již aplikace hotová a funkční, je možné ji nasadit na QA. Následuje integrace aplikace se systémy, se kterými bude kooperovat. Další fází je pak testování, které zahrnuje unit testy, systémové testy a systémově integrační testy. Po úspěšném dokončení všech předchozích testů je možné

aplikaci nasadit na UAT, kde opět proběhne testování, a sice systémově integrační testování (SIT) a uživatelské akceptační testování (UAT). Je-li aplikace úspěšně otestována, je na řadě akceptace aplikace, tedy schválení, že je aplikace v pořádku a je možné ji nasadit na produkční prostředí. Pak už je možné ji nasadit na produkční prostředí, zpřístupnit ji veřejnosti, a následně už ji jen monitorovat a udržovat v průběhu jejího používání stejně tak, jak bylo popsáno u vývoje her.

Obrázek 7 Vývoj aplikací



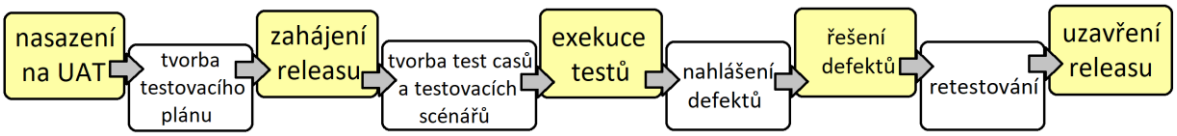
Zdroj: Vlastní zpracování

4.2.3 Průběh uživatelského testování

Uživatelské testování začíná nasazením produktu na testovací prostředí (UAT), kde je ještě před zahájením releasu. To provádí v případě externího vývoje softwaru externí dodavatel. V případě vývoje Sazkou provádí nasazení právě Sazka. Následně je vytvořen plán,

co se kdy bude testovat. Pak už může být zahájen release. Prvním krokem v releasu je vytvoření test casů a z nich případně testovacích scénářů, které si do daného releasu přiřadí test analytici. Dále je na řadě exekuce testů, která probíhá v souladu s testovacími plány. Dodržuje se především priorita jednotlivých testovacích souborů a priorita jednotlivých testovacích scénářů či test casů, které jsou v nich obsaženy. V případě identifikace případných defektů, jsou takové defekty nahlášeny kompetentním osobám a po jejich vyřešení je možné je znovu otestovat – retestovat. Po úspěšném otestování všech test casů a testovacích scénářů je možné release uzavřít a pokračovat k akceptaci produktu. Celý průběh uživatelského testování je zobrazen na následujícím obrázku.

Obrázek 8 Průběh uživatelského testování



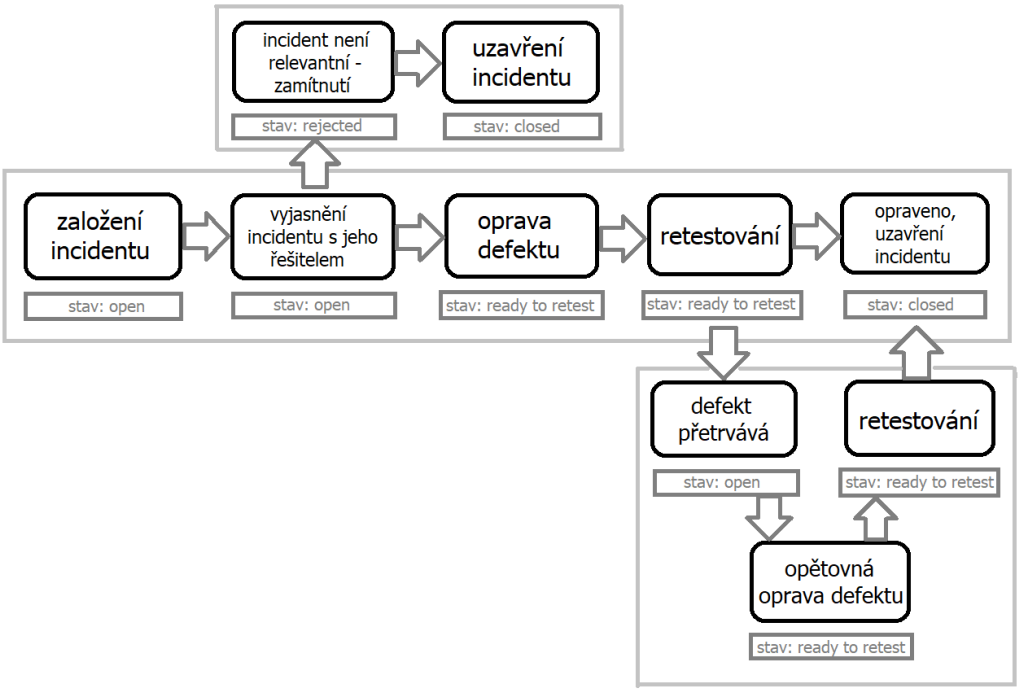
Zdroj: Vlastní zpracování

4.2.4 Životní cyklus incidentu

V případě, že je v průběhu jakékoli fáze vývoje, především ve fázi testování, zjištěn defekt, musí být zajištěna jeho dokumentace. Je třeba sepsat takzvaný incident, což je dokument, který přesně popisuje zjištěný defekt. Po založení incidentu nastává situace, kdy je potřeba v případě nejasností nebo neschopnosti řešitele nasimulovat incident, vyjasnit, jak je incident myšlen nebo jak přesně lze nasimulovat. Běžně se stává, že při simulaci rozhodují jen malé nuance, jež defekt vyvolají. Je tedy nutné velmi přesně specifikovat podmínky, za kterých byl dotyčný objekt testován a jaký postup byl dodržen. Následně je možné defekt opravit a předat zpátky na opětovné otestování neboli retestování. V případě, že je již defekt opraven, je možné incident zavřít. Když se ale defekt popsaný v incidentu neustále objevuje, je nutné vrátit ho k opětovné opravě, následně opět provést retest a po úspěšném provedení restestu je možné celý incident uzavřít.

Může nastat i situace, kdy defekt popsaný v incidentu bude zamítnut, a to z toho důvodu, že identifikovaný defekt není nežádoucím, nýbrž je popsané chování akceptovatelné či dokonce žádoucí. Celý životní cyklus incidentu je znázorněn na následujícím obrázku.

Obrázek 9 Životní cyklus incidentu



Zdroj: Vlastní zpracování

5 Současný stav testování

Testování jako takové je v Sazce vcelku novou součástí procesu vývoje. Testování přišlo s nastupující digitalizací zhruba před 3 lety. V té době byli v Sazce zaměstnaní jen 4 testeři, kteří neměli moc objektů k testování. Od té doby se začal testovací tým pomalu rozšiřovat a přibyly i testovací platformy. V současné době je testerů zhruba čtyřnásobek. Současný stav testování zahrnující například platformy, na kterých se testuje, testovací tým a software používaný pro testování bude v této kapitole detailně popsán. Ovšem budou popsány pouze ty platformy a testy, které souvisí s náplní práce test exekutorů.

5.1 Testované platformy

V současné době se provádí uživatelské testy na třech různých platformách. Nejstarší platformou jsou POS terminály (z anglického Point Of Sale), jejichž předchůdci, tedy velmi zjednodušená verze současných teminálů, jsou součástí Sazky již od 90. let 20. století. Další platformou jsou webové stránky a nejnovější platformou, která vznikla teprve nedávno, jsou mobilní aplikace. V níže uvedených podkapitolách budou detailně představeny možnosti všech tří platforem, aby bylo zřejmé, co vše je předmětem testování. V následující tabulce je zaznamenán přehled všech platforem a služeb, které na daných platformách v současné době test exekutoři testují.

Tabulka 3 Přehled testovaných objektů na platformách

	Webové stránky	Aplikace		POS terminály
		Sazka	Sazka Bet	
Technické hry	●			
Losy	●	●		
Loterie	●	●		●
Kurzové sázky	●		●	
Finanční služby				●
Mobilní služby				●

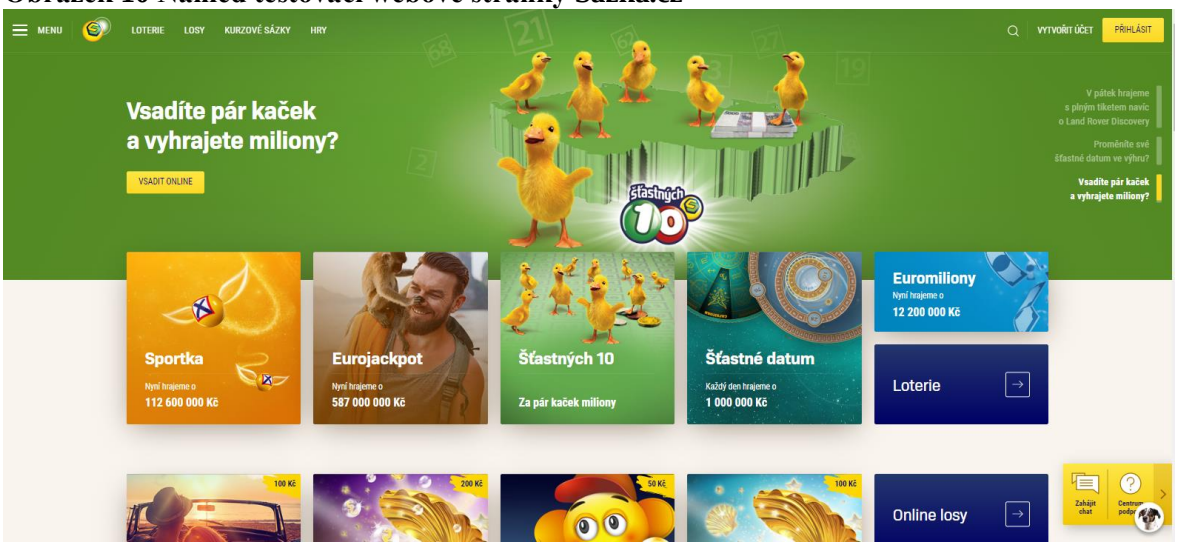
Zdroj: Vlastní zpracování

5.1.1 Webová stránka

První testovanou platformou je webová stránka, která jako jediná obsahuje téměř všechny služby, nabízené Sazkou. Tato stránka nabízí možnost vsadit si jakoukoli loterii, kurzovou sázku, koupit si losy nebo si zahrát hru. Výhodou u her a losů je, že si uživatel může vybrat, zda chce hrát za peníze nebo pouze v takzvaném „demo mode“, neboli v režimu, kdy hru či los může vyzkoušet bezplatně.

Je zde možné založit si svůj vlastní účet, díky kterému lze hrát tyto hry a losy za peníze. Zároveň to usnadňuje uživatelům vsázení loterií a kurzových sázek. Na následujícím obrázku je náhled testovací webové stránky Sazka.cz

Obrázek 10 Náhled testovací webové stránky Sazka.cz



Zdroj: Vlastní zpracování ze https://www.sazka.cz/

Webová stránka je přizpůsobena jak na počítače, tak i tablety a mobilní zařízení. Podporuje systém iOS, Android i Windows, takže je připravená pro všechny potencionální uživatele. Testuje se na webových prohlížečích Google Chrome, Firefox, Microsoft Edge, Safari a na nativních prohlížečích v mobilních telefonech.

5.1.2 Mobilní aplikace

S postupným přechodem uživatelů z počítačového prostředí k mobilním telefonům se objevila potřeba vytvořit přístup ke službám Sazky právě z mobilních telefonů. Z toho důvodu byly vytvořeny dvě mobilní aplikace. A sice Sazka a Sazkabet. Tyto aplikace ještě do budoucna doplní další dvě aplikace, které dosud nejsou spuštěné.

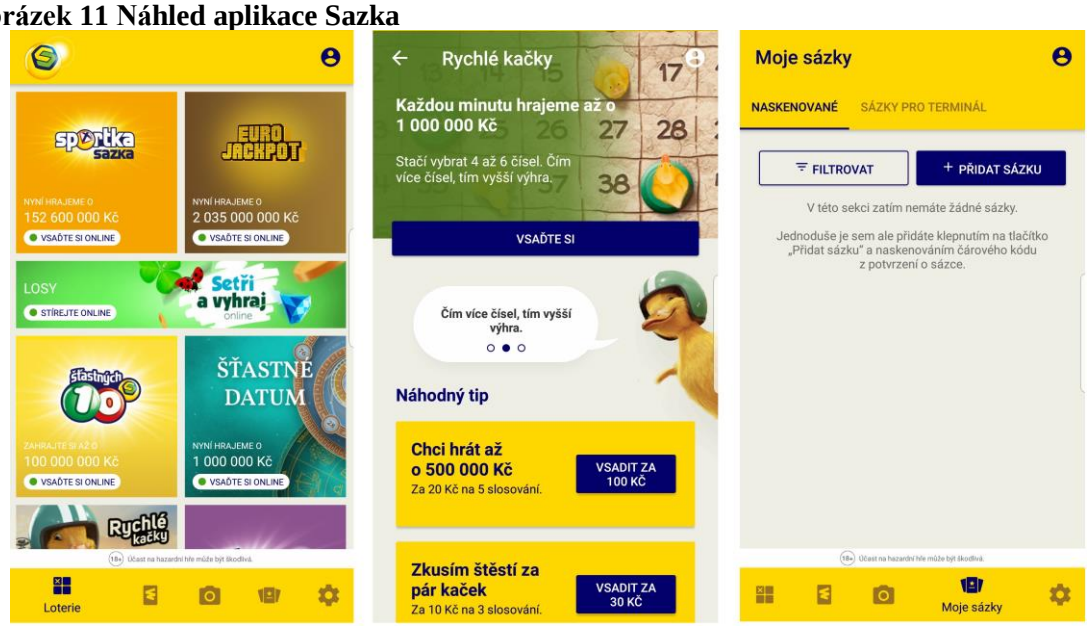
Sazka

Bezplatná aplikace Sazka, jejíž náhled je na Obrázku 11, umožňuje online sázet loterie a losy. Uživatel zde může spravovat svůj účet. Má zde přehled o své historii sázek, o aktivních sázkách – tedy dosud nevyhodnocených a může si vytvořit u jednotlivých loterií oblíbená čísla, která pak může jedním kliknutím znovu vsadit. Při blížícím se slosování vsazených loterií zašle uživateli upozornění, díky čemuž má uživatel stále přehled.

Také je možné kontrolovat sázky provedené přes terminál, a to naskenováním čárového kódu z tiketu či zadáním vybraných čísel ručně. Rovněž může vygenerovat unikátní čárový kód sázky v aplikaci pro kontrolu na terminálu. Nejbližší terminály společně s otevírací dobou lze v aplikaci taktéž vyhledat.

Uživatel se může přihlásit buď klasicky pomocí přihlašovacích údajů, které získal při založení účtu, nebo podle pin kódu, Touch ID či Face ID. Sazka aplikace je určena pro telefony s OS Android verze 5 a vyšší a lze ji stáhnout na webových stránkách nebo si odkaz na aplikaci nechat poslat pomocí SMS zprávy. Pro iOS je dostupná na App Store a to pro verze iOS 10.3 a vyšší.

Obrázek 11 Náhled aplikace Sazka



Zdroj: Vlastní zpracování z aplikace Sazka

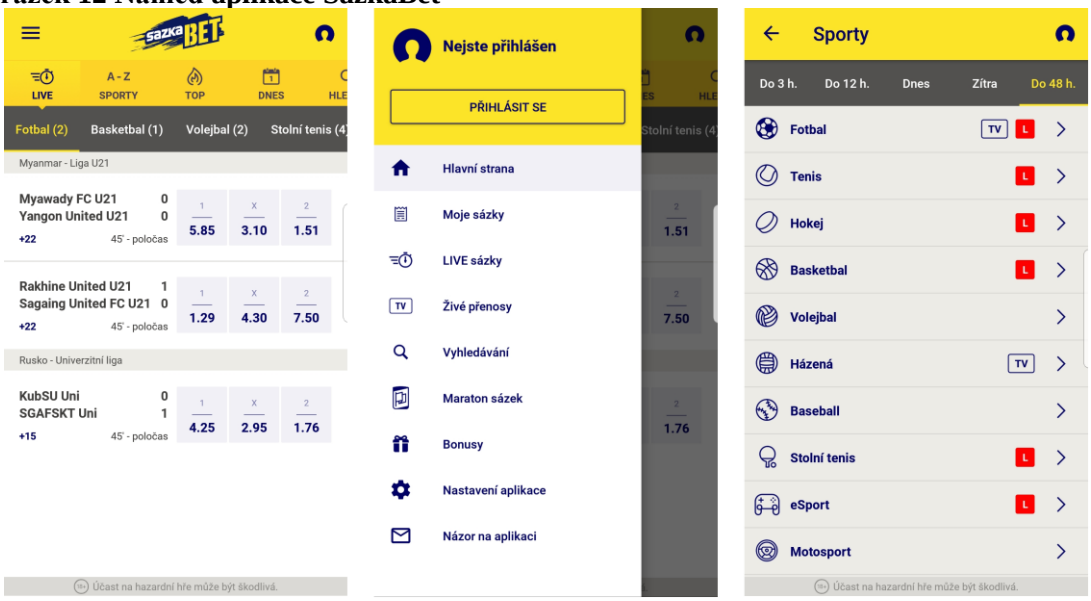
Sazka BET

Tato aplikace je určená pouze pro kurzové sázky. Nabízí rozsáhlé spektrum sázek na sportovní zápasy a sportovní výkony vybraných sportovců. Nabízí Live Stream

vybraných zápasů a v případě, že není k dispozici, je možné sledovat zápas díky vizualizaci zápasů, kterou aplikace nabízí. Je možné sázet sázky před zápasem a LIVE sázky, které lze kombinovat na jeden tiket, a rychlé sázky. Uživatel může přehledně sledovat historii svých sázek a detailní statistiky všech zápasů. Kromě zmiňovaných možností chystá Sazka řadu nových funkcí, které aplikaci rozšíří.

Sazkabet je stejně jako aplikace Sazka určena pro OS Android verze 5 a vyšší a pro iOS pro verze iOS 10.3 a vyšší, je zcela zdarma a je taktéž dostupná na App Store a webových stránkách Sazky. Na rozdíl od aplikace Sazka ale není možné se přihlásit pomocí pin kódu, jen pomocí přihlašovacích údajů, Touch ID či Face ID. Náhled aplikace Sazkabet je na následujícím obrázku.

Obrázek 12 Náhled aplikace SazkaBet



Zdroj: Vlastní zpracování z aplikace SazkaBet

5.1.3 POS terminály

Zákazníci mohou využít služeb Sazky nejen online, ale také na více, než 7000 terminálech po celé České republice, které jsou obsluhovány proškoleným personálem. Tradiční službou, nabízenou terminály je podání či zrušení jakékoli loterie a následně výběr případné výhry. Další službou terminálů je možnost si dobít kredit na předplacenou kartu Sazka mobil. Při dobíjení je možné využít Sazka mobil šance a hrát tak o jackpot a zároveň je možné si vybrat jednu ze čtyř odměn ke kreditu při dobití nad 300 Kč. Dobíjet lze

i v rámci tří největších mobilních operátorů v České republice a jejich virtuálních operátorů, ale pouze kredit bez výhod.

Přes tyto terminály lze také platit faktury, provádět online platby pomocí služby Paysafecard a klienti Air Bank mohou dokonce vybírat hotovost prostřednictvím terminálů ze svého bankovního konta.

5.2 Testovací zařízení

Aby mohly být všechny objekty dobře otestovány, je třeba vyzkoušet jejich funkčnost na co nejširším spektru zařízení. Proto má Sazka tolik zařízení, aby pokryla všechny možnosti, které mají běžní uživatelé. To znamená, že má osm stolních počítačů, kde jsou různé typy Windows, jeden Macbook a tři notebooky také s operačním systémem Windows. Z mobilních zařízení je k dispozici pět iPhonů, každý s jiným iOS a deset telefonů různých značek, které mají dohromady všechny aktuálně dostupné verze operačního systému Android. Disponuje i dvěma tablety, jedním s operačním systémem iOS a druhý s Androidem. Zařízení, ke kterému se běžný uživatel nedostane, jsou terminály POS, kterých je testerům k dispozici celkem pět. V následující tabulce je sepsán přehled všech zařízení.

Tabulka 4 Soupis testovacích zařízení

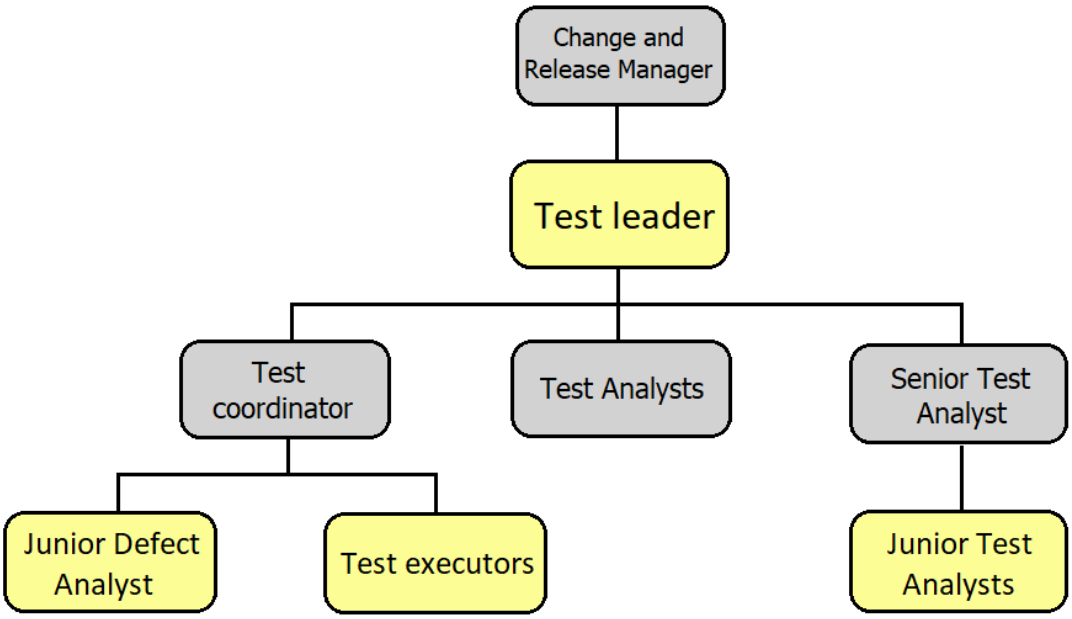
Testovací zařízení	Počet (ks)	Značka	Operační systém
Mobilní zařízení	3	Samsung	Android 7 a 9
	2	Xiaomi	Android 7.1.2; 9
	2	Honor	Android 9
	5	iPhone	iOS 12.2; 13.1.1; 13.1.2; 13.2.2; 13.3.1
	1	Huawei	Android 8
	1	LG Nexus	Android 8.1
	1	GooglePixel	Android 10
Tablety	1	Lenovo	Android
	1	iPad	iOS
Stolní počítače	8	HP	Windows 7; 8.1; 10
Notebooky	3	HP	Windows
	1	Mackbook	iOS
POS terminály	5	neuvedeno	neuvedeno

Zdroj: Vlastní zpracování

5.3 Velikost a složení týmu – organizační struktura

Testovací oddělení má dohromady zhruba 26 zaměstnanců. Tyto počty se neustále mění. Je to dáno nestálostí zaměstnanců, kteří nepracují na plný úvazek a také tím, že jsou výjimečně najímání externí testeři. V čele celého oddělení je Change and Release Manager, jehož podřízeným je Test leader. Ten řídí Test coordinatora, Test Analytiky a Senior Test Analytika, který má pod sebou Test Analytiky na juniorské úrovni. Test coordinator vede tým Test exekutorů a Junior Defekt Analytika. Schéma organizační struktury testovacího oddělení je vyobrazeno v následujícím obrázku. Dále budou popsány jednotlivé pozice s detailnějším zaměřením na Test exekutory, kteří jsou pro účely této práce stěžejní.

Obrázek 13 Organizační struktura testovacího oddělení



Zdroj: Vlastní zpracování

Change and Release Manager

Na vrcholu organizační struktury testovacího oddělení je Change and Release Manager, který má na starosti vedení celého oddělení. Řídí proces změn v systémech a aplikacích a je zodpovědný za vyhodnocení dopadů těchto změn. Všechny změny vyjednává a koordinuje s dodavateli. Sestavuje a řídí release plány. Řídí nasazení změn na testovací i produkční prostředí.

Test leader

Přímým podřízeným Change and Release Managera je Test Leader. Zodpovídá za kvalitu testování, nastavuje a kontroluje procesy a řídí Test coordinatora, Test Analytiky a Senior Test Analytiky. Připomínkuje Change and Release Managera.

Test coordinator

Náplní práce Test coordinatora je reportování stavu exekuce testů, dále vyřizování administrativy a koordinace exekuce testů a restestů defektů. Zodpovídá za řízení týmu externích IT testerů a za zajištění a správu testovací infrastruktury jak v rámci Sazky, tak i s externími dodavateli. Řídí a koordinuje Test exekutory a Junior Defekt Analytika.

Test Analysts, Senior Test Analysts, Junior Test Analysts

Pozice test analytika je rozdělena do tří úrovní a sice Test Analytik, Senior Test Analytik a Junior Test Analytik. Rozdíl mezi těmito úrovněmi je ve zkušenostech analytiků a dovednostech, kterými disponují. Na nejvyšší úrovni jsou Test Analytici, dále jsou Senior Test Analytici a nejméně zkušenými analytiky jsou Junior Test Analytici, kteří většinou nemají moc předchozích zkušeností s touto prací. Dále se bude hovořit o test analyticích obecně, nebudou rozlišovány jednotlivé úrovně.

Náplní jejich práce je na základě dokumentace spojené s testovaným objektem navrhnout a vytvořit test casy. Test casy musí obsahovat návody na všechny možné funkce a kombinace funkcí, které testovaný objekt nabízí. Nejen že musí napsanými test casy zajistit otestování všech funkcí a jejich kombinací, ale musí také myslet na to, aby byl testovaný objekt otestován na všech typech testovacích zařízení. Následně testům přiřadí jejich priority. Důležitou činností je i konzultace případných nejasností s Test exekutory, ať už se jedná o nejasnosti v test casu nebo o konzultování potencionálních defektů.

Junior Defect Analyst

Junior Defect Analytik kontroluje správnost zadávání incidentů, kontroluje životní cyklus incidentů a eskaluje tyto incidenty na Test Coordinatora. Dále provádí manuální synchronizaci chyb mezi bug trackingovými nástroji, tedy mezi softwarem pro podporu testování SpiraTest a softwarem pro podporu testování, který využívají externí dodavatelé.

Test executor

Test exekutoři provádí testy podle předem sepsaných test casů nebo testovacích scénářů. Identifikované defekty reportují formou sepsání incidentu. Konzultují chování testovaného softwaru s test analytiky. Testy provádí pouze manuální, na úrovni systémově integračních testů (SIT) a uživatelských akceptačních testů (UAT). Většinou se jedná o testování v rámci webové stránky a aplikací.

Na pozici test exekutor (dále jen tester) pracuje v současné době 15 zaměstnanců. Tým se skládá ze studentů vysokých škol, kteří jsou zaměstnání na dohodu. Svůj čas v práci si mohou naplánovat podle svých možností. Vždy však minimálně pět pracovních dní dopředu přes rezervační systém SuperSaaS. Díky tomu má test coordinator přehled, kolik testerů bude ten den k dispozici a může tak plánovat práci na daný den přesně pro konkrétní testery. Náplň jejich práce se odvíjí od zadání, které dostanou každé ráno.

Výstupy z jejich práce si navzájem nijak nepředávají, pouze zaznamenávají výsledky do softwaru SpiraTest.

5.4 Plánování testování

Testování probíhá po jednotlivých release. Tedy po souborech testovacích scénářů nebo jednotlivých test casů, rozřazených do složek, náležející k nově vyvinutému softwaru, jeho části nebo nové funkci, která je připravena k otestování. Jejich pořadí je určeno třemi faktory, a to termínem nasazení, termínem sepsání potřebných test casů a testovacích scénářů, které závisí na unikátnosti a složitosti testované funkce či objektu, a na termínu, do kdy má být celý release otestován.

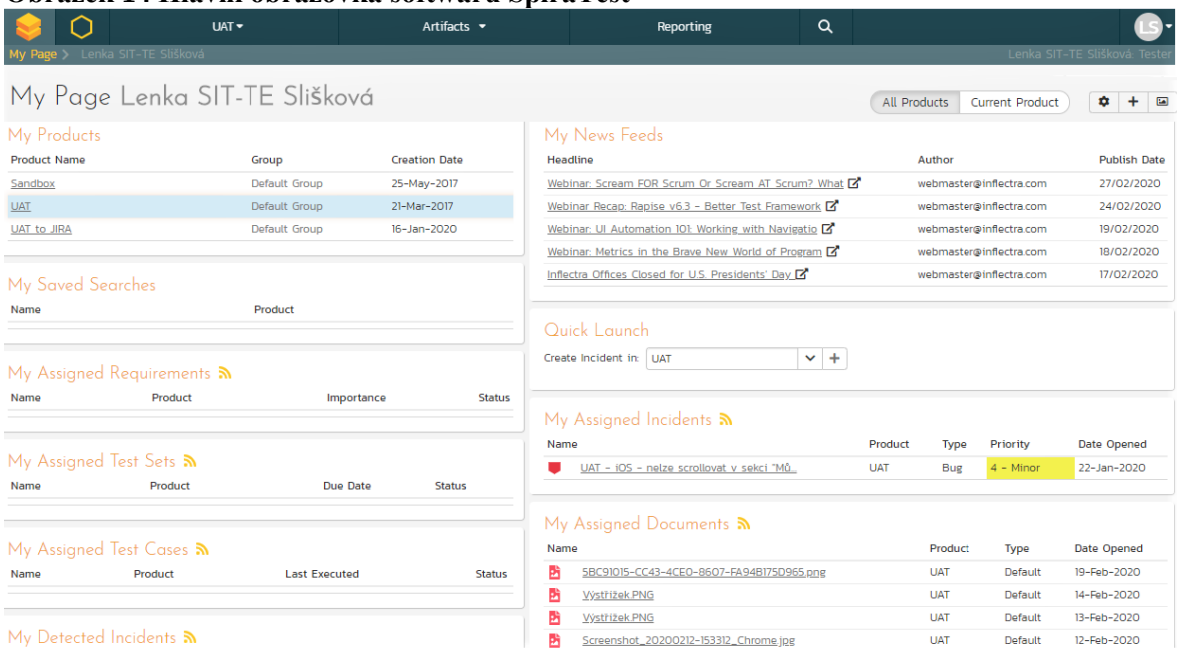
Release a jeho konkrétní složky s test casy se testerům přiřazují pravidelně každé ráno. Prostřednictvím komunikačního kanálu Slack rozvrhne test manager testy pro všechny testery, kteří jsou ten den přítomni v práci. V některých případech rozdělují práci i test analytici, kteří si vybírají testery pro jimi napsané testy. Typy testů jsou většinou přiřazovány dle schopností testerů. Výjimečně jsou přiřazeny náročnější úkoly méně zkušeným testerům, a to z důvodu získání nových zkušeností.

Testování se tedy plánuje na každý den zvlášť s tím, že se bere ohled na již zmíněné tři faktory.

5.5 Software SpiraTest

Jako platforma pro podporu testování slouží Sazce software SpiraTest od společnosti Inflectra. Jedná se o platformu pro správu test casů, testovacích scénářů a incidentů s popsanými defekty. Je to nástroj, který je kompletně dostupný z webové stránky, tudíž není potřeba stahovat do všech zařízení nový software. Stačí mít unikátní URL adresu vygenerovanou pro danou firmu či jiný objekt, který si tento software předplatil. Každý zaměstnanec získá své přihlašovací údaje, pod kterými se může přihlásit odkudkoli. Po přihlášení se objeví hlavní obrazovka – tzv. „My Spira“, kde má zaměstnanec svůj profil. Může zde například sledovat incidenty, které založil, incidenty, které mu byly přiděleny na opětovné otestování, zprávy o novinkách nebo rozpracované testy.

Obrázek 14 Hlavní obrazovka softwaru SpiraTest



Zdroj: Vlastní zpracování ze softwaru SpiraTest

5.5.1 Způsob testování ve SpiraTest

Testy jsou uspořádány do jednotlivých testovacích sad – tzv. release. Release vždy obsahuje několik složek, ve kterých jsou pak test casy. Pro snadné dohledání má každý test case svoje unikátní číslo a název.

Test casy pak obsahují dílčí kroky – tzv. stepy, ve kterých je popis toho, co přesně má tester dělat pro otestování daného objektu. Popsán je i očekávaný výsledek či chování

objektu. V případě, že výsledkem testování je očekávaný výsledek, označí tester příslušný step jako „pass“ – tedy, že prošel v pořádku. Jako „caution“ je označován step, ve kterém je nalezen defekt. Na každý nový defekt je potřeba sepsat nový incident, následně připojit jeho odkaz do záložky incidentů příslušného kroku a napsat číslo incidentu, které se automaticky generuje, do komentáře test casu. Také je možnost, že je již takový incident založen, což znamená, že je potřeba pouze připojit incident do záložky incidentů a napsat jeho číslo do komentáře. Když najde tester defekt, který brání dalšímu pokračování v testování, označí příslušný krok jako „blocked“ a připíše do komentáře, z jakého důvodu nelze pokračovat dál, a případně opět sepíše incident nebo připojí již existující.

5.5.2 Incidenty

V případě, že se při testování objeví jakýkoli rozpor s testovacím scénářem nebo nastane nestandartní chování, je třeba zjistit, zda se opravdu jedná o chybu nebo jestli je dané chování v pořádku. Může nastat i situace, kdy se bude jednat o chybu, která ale bude akceptovatelná a nebude třeba k ní sepisovat incident. Z toho důvodu před založením nového incidentu diskutuje tester danou situaci s test analytikem.

Incidenty se dělí dle priority do tří kategorií, a sice minor, major, critical. Incident označený jako minor je takový incident, který není závažný a nijak nebrání fungování testovaného objektu. Může se jednat například o chybu v textu či chybu v grafice. Jako major je označován incident, který je vážnější, ale stále zachovává základní funkčnost testovaného objektu. Do této kategorie spadá například chybějící historie her, nefungující limity zodpovědného hraní nebo třeba chybějící herní bilance uživatele. Nejzávažnější je incident s prioritou critical. Takový incident zabraňuje fungování testovaného objektu nebo ho velice vážně narušuje. Jedná se o situaci, kdy například nelze spustit hru, je odečítána z uživatelského konta jiná částka, než je deklarovaná ve hře, nebo se naopak nepřičítá výhra na konto uživatele.

5.6 Typy prováděných testů

V procesu testování je prováděna celá řada testů. Ze strany testerů jsou prováděny pouze uživatelské testy. Každý test má svou prioritu, což určuje pořadí, ve kterém jsou testy prováděny. Je rozlišováno pět prioritních úrovní. Nejvyšší priorita je „critical SMOKE“. Účelem testů s touto prioritou je zjistit, zda u testovaného objektu fungují alespoň základní

funkce. Jako třeba otevření testované hry a správný odečet vsazené částky z uživatelského účtu. Další prioritou je „critical“, což označuje testy, které musí být prováděny přednostně. Dále je priorita „high“, „medium“ a „low“. Testy s prioritou „low“ se provádí až jako úplně poslední.

Následně budou popsány jen ty testy, které jsou prováděny testery. Testeři provádí pouze manuální testy, tedy testují vždy podle test casů či testovacích scénářů, které simulují různé aktivity či přesně daný sled aktivit, jenž může v praxi nastat. Ověřují software z pohledu zákazníka.

5.6.1 Smoke testy

Jako první se spouští takzvané smoke testy. Jsou vždy označeny prioritou „critical SMOKE“. Jde o krátké testy, které ověří, zda je testovaný objekt základně funkční a lze ho dál testovat. Tedy, že základní či klíčové funkce bezchybně fungují.

5.6.2 Funkční testy

O poznání důkladnější jsou testy funkční. Jejich úkolem je zjistit, zda všechny funkce testovaného objektu pracují tak, jak je popsané v test casu. To znamená detailně otestovat všechny kombinace a funkce testovaného objektu a zjistit, zda funguje bez problému. Funkční testy se především provádí u testovaní technických her, losů a loterií. S ohledem na testovaný objekt se často dělí na tři skupiny, a to quick test, all test a general test.

Quick test

Účelem Quick testu je rychlé otestování testovaného objektu. Provádí se na velkém množství testovacích zařízení, protože je nutné zjistit, zda funguje na všech internetových prohlížečích, na všech verzích a typech operačních systémů a na všech typech obrazovek. Slouží tedy pro rychlé ověření funkčnosti na všech typech zařízení a operačních systémů.

All test

All testy se provádí v menším rozsahu. Vždy alespoň na jednom mobilním zařízení s operačním systémem Android, na mobilním zařízení s iOS a na dvou počítačích s různou verzí Windows nebo iOS. Tyto testy jsou rozsáhlé. Mají za úkol důkladně otestovat všechny funkce objektu, různé kombinace těchto funkcí a podmínky, které lze u některých objektů nastavit.

General test

General testy se zaměřují především na testování časových limitů. Jde například o to, že v případě hraní technických her je po dvou hodinách hraní povinná pauza 15 minut, kterou je třeba dodržet tak, že hráč nebude moct po tuto dobu pokračovat v hraní.

5.6.3 Testování dokumentace – Gameplan

Tento způsob testování se využívá především pro technické hry. Kontroluje se správnost výše výher a odečtu proher z uživatelského účtu a soulad součtu kombinací výher s výherní tabulkou. Důležitým prvkem je i detailní kontrola souladu všech textů ve hře a v Gameplan.

5.6.4 Konfirmační testování – retestování

Jedná se o testování založených incidentů. Incidenty, které již byly opraveny je třeba znovu otestovat – retestovat a prověřit, zda daná komponenta již funguje dle očekávání. Toto testování může být u jednoho incidentu provedeno i vícekrát, a to v případě, že se defekt popsaný v incidentu nepodaří úspěšně opravit.

Retestování nemá žádný harmonogram ani prioritu. Testeři si většinou musí sami hlídat, zda na ně nebyl přiřazen některý incident k retestování. V případě, že se nahromadí větší množství neodbavených incidentů připravených k retestování, je upozorněn testovací tým, aby tyto retesty provedl.

5.6.5 Regresní testování

Regresní testování je prováděno po dokončení releasu nebo po určité skupině dokončených releasů, které spolu souvisí. Jeho úkolem je ověření, že provedené změny nebo implementace nových funkcí nemělo žádný vliv na stávající funkce a vlastnosti. Testují se již úspěšně otestované objekty, tedy ty, které nebyly změněny v programovém kódu. Je totiž možné, že se u některých objektů objeví nové chyby, které před tím nebyly identifikovány nebo se znovu objeví ty, které již byly opraveny.

6 Identifikace slabých míst spojených s uživatelským testováním

V následující kapitole budou podrobně popsány problémy a slabá místa, která se vyskytují v testovací fázi a jsou spojena s uživatelským testováním. Budou identifikována jen ta slabá místa, která přímo souvisí s prací testerů.
Slabá místa byla identifikována na základě popisu současného stavu testování, dále na základě diskuse s testery, kteří v Sazce pracují, a pomocí pozorování chodu testovacího oddělení.

6.1 Absence školení

Problémem, který je zdánlivě nepodstatným, je absence jakéhokoli školení. Tester je pouze seznámen se softwarem SpiraTest, kde jsou mu vysvětleny základní funkce, jak se testuje podle test casů a testovacích scénářů a jak se zakládají nové incidenty. Dále je seznámen s testovací verzí webové stránky Sazky a jejím obsahem.
Jediné školení probíhá tím způsobem, že si nově zaměstnaný tester sedne vedle jednoho ze zkušenějších testerů a sleduje ho při práci. Kdo bude testování předvádět a vysvětlovat všechny postupy se vybírá podle toho, kdo zrovna testuje technické hry. To znamená, že není nikdo speciálně vyškolený na to, aby nové testery zaučil. Takové zaučení probíhá jeden až dva dny, pak už testuje každý sám. Tím vzniká situace, že se mohou předávat zkreslené informace, a tester fixuje chybné postupy. Zároveň za tak krátkou dobu nelze předat všechny důležité informace, v důsledku čehož se musí nový tester velké množství doučit a dohledat sám nebo se doptávat kolegů.

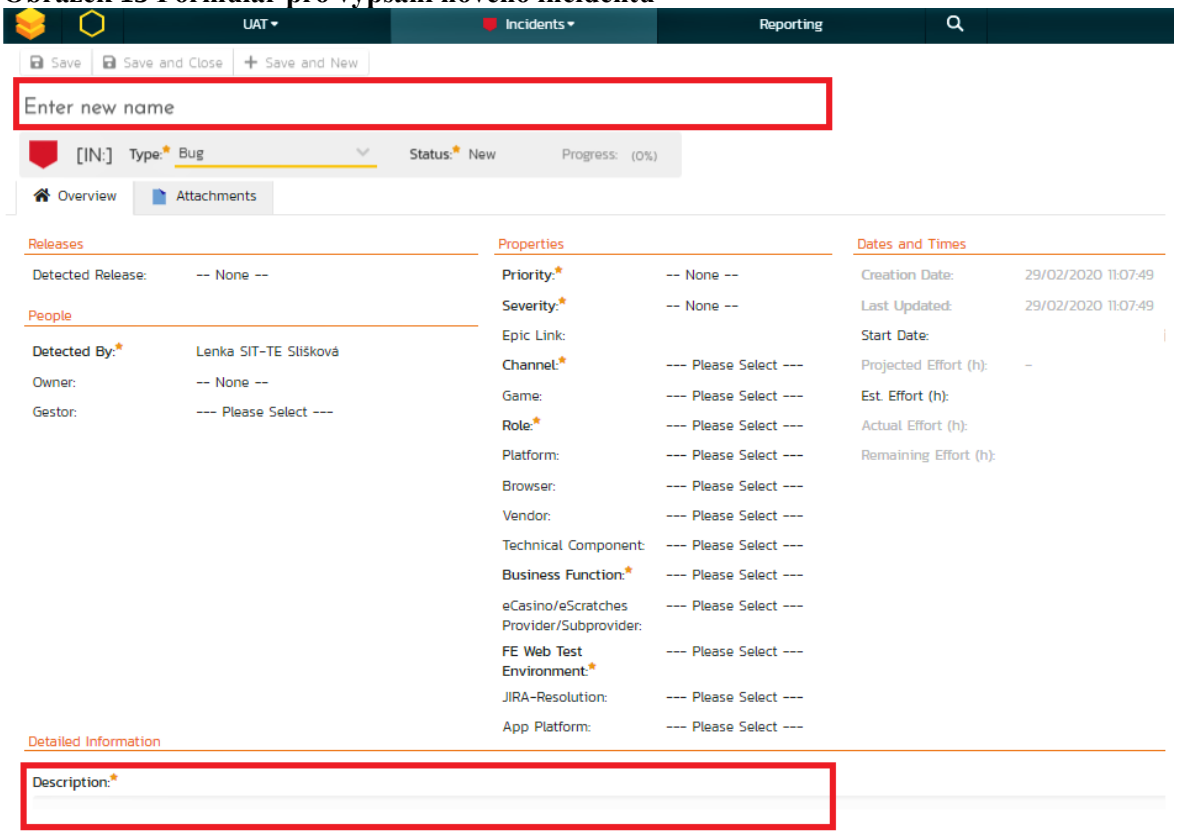
6.2 Zápis incidentů

Na zápis incidentů neexistuje jednotná forma. V softwaru SpiraTest je na zápis incidentů sice formulář, ale není zcela jednotná forma, jak ho vyplnit. Některé položky lze vyplnit vybráním ze seznamu možností. Jako například priorita, business funkce či vlastník incidentu. Problém nastává v případě vyplnění názvu incidentu a jeho popisu, které jsou na Obrázku 15 označeny červeným obdélníkem.
V názvu se vypisuje, jaké oblasti se incident týká (aplikace, losy, zodpovědné hraní...), dále zařízení, na kterém se incident objevuje (mobilní zařízení, počítač či POS terminály),

operační systém, prohlížeč, případně i ID hry či losu a velmi stručný popis incidentu. Co se týče hlavního popisu incidentu, měl by zde být uveden přesný popis toho, jaké je očekávané chování dané komponenty, jaké je reálné chování a scénář s popisem, jak nasimulovat situaci, kdy dochází k incidentu. Je vhodné také uvést výchozí podmínky testování, což znamená zapsat přihlašovací údaje uživatelského účtu, na kterém byl incident detekován, napsat přesný typ testovacího zařízení či aktuální verzi operačního systému.

Problémem je, že není nikde sepsána jednotná forma pro zápis názvu a popisu incidentů a také to, co by měl takový název a popis incidentu všechno obsahovat. To způsobuje nejasnosti při komunikaci s řešiteli incidentů a velmi zdržuje práci, protože může takové vyjasnění incidentu vyžadovat delší komunikaci a tím může být pozdržen i celý release. Někdy také nejasný popis způsobí, že jiný tester zadaný incident nepochopí správně a vytvoří nový incident na tentýž defekt, čímž vzniká jejich duplicita.

Obrázek 15 Formulář pro vypsání nového incidentu



Zdroj: Vlastní zpracování ze softwaru SpiraTest

6.3 Absence testovacího plánu

Slabou stránkou v procesu testování je i absence kompletního testovacího plánu. Chybí například soupis všech typů prováděných testů, popis testovaných oblastí a testovací strategie. Neexistuje žádný dokument, kde by byl sepsán jasný přehled a mohl do něj kdokoli nahlédnout.

6.4 Jednotné testovací scénáře

Z důvodu vysokého počtu nových technických her je třeba psát v krátkém časovém úseku velké množství test casů a tvořit z nich testovací scénáře. To je řešeno využitím již předepsaných test casů – šablon, které lze opakovaně využít. Jelikož je většina her z velké části založena na stejném principu, je tento způsob psaní výhodný. Problém přichází v případě, že má výrobce hry jiné požadavky, než je uvedené v test casu. V takovém případě takovou odlišnost od test casu vyhodnocuje tester jako chybu a kontaktuje analytika, aby s ním danou situaci konzultoval, což zabírá čas jak testerovi, tak i test analytikovi. Nebo přímo založí nový incident, který bude následně zamítnut, což zabere čas dalším zaměstnancům. V obou případech je zřejmé, že taková situace zdržuje vždy minimálně dva zaměstnance od práce a snižuje se tím tak jejich výkonnost. Navíc se běžně stává, že si testeři nepředají informaci a pak opakují tytéž dotazy směrem k test analytikům jako již dříve jejich kolegové.

6.5 Nedostatek testovacích zařízení

I když je dostupná široká škála testovacích zařízení, stává vcelku se běžně, že tester nemá k dispozici zařízení, na kterém potřebuje testovat. Je to dáno tím, že u většiny testů jsou přímo deklarovaná zařízení, na kterých je příslušný test nutné provést. To znamená, že tester nemůže vzít jiné zařízení a musí čekat, až bude dané zařízení volné. Tento nedostatek se vyskytuje především u Apple produktů. Nejčastější problém se vyskytuje u Macbooku a u iPhonů.

Úskalím je i skutečnost, že jsou testovací terminály v jiných částech budovy, než mají testeři svou kancelář a nemají je u sebe. To znamená, že při každém jednorázovém úkonu na terminálu musí tester projít část budovy a následně se opět vrátit zpět, což vcelku zdržuje při práci.

6.6 Výkonnost testerů

Výkonnost testerů je monitorována pomocí výsledků ve SpiraTest. Sleduje se čas strávený ve SpiraTest, počet hotových stepů, počet dokončených test casů, počet dokončených testovacích scénářů a počet nově založených incidentů. To ovšem není zcela dostačující. Výsledky těchto sledování sice dají hrubou představu o výkonnosti, ale ne zcela přesnou a ne zcela relevantní.

Tester může mít velké množství dokončených test casů, ale za cenu, že je otestoval nekvalitně a přehlédl některé defekty. Proto je velmi obtížné měřit výkonnost testerů a ti toho mohou následně zneužívat. Někteří testeři mají pak tendence svojí práci odbývat nebo pracovat málo.

6.7 Nedostatečná komunikace a informovanost

V případě, že je nalezen při testování defekt, je zapsán do systému SpiraTest a testeři ho mohou případně dohledat. Tudíž vědí, že je defekt již reportován a připojí jeho ID do právě prováděného testu. Pokud však je identifikováno chování softwaru, které se může zdát chybové a po konzultaci s test analytikem se ukáže, že takové chování je v pořádku nebo je to defektem, který ale nebude řešen, nikam se nezaznamenává. To znamená, že se pak každý tester zabývá stejným problémem a plýtvá jak svým časem, tak časem test analytika.

7 Návrh na změnu

V této kapitole bude sepsán návrh činností, které mohou vést ke zlepšení či k zefektivnění dosavadního procesu testování v souvislosti s identifikovanými slabými místy. Některé návrhy řeší dokonce více slabých stránek najednou. Jednotlivé návrhy řešení byly sestaveny tak, aby s co nejnižší investicí, a to jak časovou, tak i finanční, bylo dosaženo co největších pokroků.

Přehled identifikovaných slabých míst a k nim příslušných návrhů je sepsán v následující tabulce:

Tabulka 5 Slabé stránky a návrhy jejich řešení

Číslo kapitoly	Slabá stránka		Číslo kapitoly	Návrh řešení
6.1	Absence školení	→	7.1	Důkladné zaškolení a školicí manuál
6.2	Zápis incidentů			
6.3	Absence testovacího plánu			
6.4	Jednotné testovací scénáře	→	7.2	Vytvoření sdíleného dokumentu
6.5	Nedostatek testovacích zařízení	→	7.3	Nákup a přesun testovacích zařízení
6.6	Výkonnost testerů	→	7.4	One to one meeting
6.7	Nedostatečná komunikace a informovanost	→	7.5	Informační tabule

Zdroj: Vlastní zpracování

7.1 Důkladné zaškolení a školicí manuál

- Slabé místo: absence školení, zápis incidentů, absence testovacího plánu

Oblast, kde je značný prostor pro zlepšení, je školení a školicí manuál, které v současné době zcela chybí. Školení sice v lehké formě, která byla zmíněna v předešlé kapitole, existuje, ale je nedostatečná. Pro porozumění testerů všem procesům a tím i pro kvalitněji odvedenou práci je důkladné seznámení se systémem a náplní práce velice důležité. Investice do školení tohoto typu by nebyla vysoká, ale může mít velký vliv na efektivitu práce budoucího testera.

Následně bude detailně popsána forma a náplň školení a obsah školicího manuálu, který by měl být vytvořen.

7.1.1 Školení testerů

Školení by mělo probíhat alespoň dva dny, ale lze ho uzpůsobit dle schopností a zkušeností školeného testera. Hlavní myšlenkou je důkladně projít všech deset oblastí, které jsou rozepsány v Tabulce 6 tak, aby jim školený tester rozuměl a byl schopen nabyté znalosti aplikovat v praxi. Protože nezbytnou součástí školení je kontrola porozumění, je v harmonogramu uveden sloupec „splněno“. Dokud nebude ověřeno, že nový tester prošel úspěšně danou oblastí školení, nemůže být pole „splněno“ zaškrtnuto a školení nemůže být ukončeno. Pro dokončení školení je potřebné mít všechny oblasti splněné.
V Tabulce 6 je sepsán návrh harmonogramu školení, rozdělený do dvou částí, kde každá část by měla ideálně představovat jeden den. Ve druhém sloupci je název oblasti, které se školení týká, a ve třetím sloupci je popis toho, co by mělo obsahovat. Harmonogram školení byl vytvořen s ohledem na návaznost jednotlivých oblastí. Témata školení byla vybrána tak, aby zahrnovala všechny oblasti testerovy práce. To znamená, že po absolvování školení bude nový tester schopný fungovat jako plnohodnotný člen týmu.

Tabulka 6 Návrh harmonogramu školení

HARMONOGRAM ŠKOLENÍ			
1. část			splněno
1	**Seznámení**	Představení nově příchozího testera ostatním kolegům a seznámení se s pracovním prostředím.	
2	**Úvod**	Náplň práce test exekutora, organizační struktura.	
3	**Založení účtu**	Založení a plné ověření uživatelského testovacího účtu.	
4	**Slack**	Založení účtu, přihlášení do firemního Slacku.	
5	**SpiraTest**	Získání přihlašovacích údajů. Seznámení se SpiraTest a ukázka způsobu provádění testů podle test casu.	
6	**Incidenty**	Struktura psaní incidentů a jejich náležitosti.	

2. část			splněno
7	**Typy prováděných testů**	SMOKE testy, funkční testy, retestování, testování dokumentace a regresní testování.	
8	**Testované platformy**	Webové stránky, aplikace Sazka, aplikace SazkaBet, POS terminály.	
9	**Testované produkty**	Technické hry, losy, loterie, kurzové sázky, finanční a mobilní služby.	
10	**NeoSphere**	Práce s uživatelským účtem a další nejčastěji používané funkce.	

Zdroj: Vlastní zpracování

Školitelé

Školení by mělo probíhat pod vedením zaměstnance, který je k tomu předem určen, tedy pod vedením kompetentní osoby. Rozhodně by se nemělo jednat o nahodilého testera. Měl by proběhnout výběr alespoň dvou zaměstnanců, kteří pracují ve společnosti nejméně půl roku. Ideálně by se mělo jednat o dva testery a jednoho test analytika, a to z důvodu, že testeři jsou nasmlouvaní jen brigádně a může se stát, že v určitý den nebude přítomen ani jeden z nich. V tom případě bude v záloze jeden test analytik, který bude moct roli školitele převzít.

Takto vybraní zaměstnanci by měli sami projít školením, které bude zahrnovat instrukce, jak přesně by školení mělo probíhat a co by mělo být jeho obsahem. Školení pro ně připraví Test leader. Také by bylo vhodné, aby si každý školitel důkladně pročetl manuál a případně doplnil o užitečné informace nebo se doptal na nejasnosti, díky čemuž by se předešlo neschopnosti školitele vysvětlit nějakou z části manuálu novému testerovi.

7.1.2 Školicí manuál

Značnou pomůckou pro školení by byl školicí manuál. Takový manuál by sloužil jako podklad ke školení. Navíc by takový manuál následně zůstával nově příchozímu testerovi a současně i již déle pracujícím testerům neustále k dispozici. Obsahem by mělo být nejen to, jak například fungují testované platformy, ale i základní organizační struktura, popis SpiraTest a její možnosti a další užitečné informace.

Školicí manuál by také obsahoval popis náležitostí a struktury incidentu, jejichž psaní je taktéž jednou z identifikovaných slabých stránek.

Zároveň by sloužil jako dobrý základ pro tvorbu kompletního testovacího plánu, který v Sazce v současné době chybí.

V následující tabulce je soupis kapitol, který by měl školicí manuál obsahovat. Zahrnuje sedm stěžejních kapitol s několika podkapitolami. Ve třetím sloupci tabulky je navíc v bodech sepsán obsah jednotlivých kapitol.

Tabulka 7 Obsah školicího manuálu

ŠKOLICÍ MANUÁL PRO TEST EXECUTORY		
Číslo kapitoly	**Název kapitoly**	**Obsah**
1	Úvod	• Náplň práce testera • Jak by měl pracovat • Jak naopak nepracovat
1.1	Komunikace	• Messenger Slack • Komunikační skupiny
1.2	Organizační struktura	• Schéma testovacího oddělení • Jmenný soupis Test Analytiků s popisem jejich oblasti působení, v rámci které píšou test casy
2	Testované platformy	Krátký úvod, zbytek bude řešen v podkapitolách níže
2.1	Webové stránky	• Představení webové stránky • Založení uživatelského testovacího účtu • Ověření uživatelského testovacího účtu
2.2	Mobilní aplikace	• Představení aplikace Sazka • Představení aplikace SazkaBet • Návod na aktualizaci verze testované aplikace (iOS + Android)
2.3	Terminály POS	• Představení terminálu POS • Nejčastěji využívané funkce
3	SpiraTest	• Popis softwaru SpiraTest • Složky, se kterými se pracuje • Používané funkce
3.1	Test case	• Spuštění • Čtení jednotlivých stepů • Vyhodnocení

3.2	Incidenty	• Soupis všech náležitostí incidentu • Formát psaní incidentu
4	Typy prováděných testů	• Smoke testy • Funkční testy (quick, all general) • Testování dokumentace • Retestování • Regresní testování
5	Backend – NeoSphere	• Popis softwaru NeoSphere • Složky, se kterými se pracuje • Používané funkce
6	Testované produkty	• Technické hry • Loterie • Losy • Kurzové sázky • Terminálové produkty
7	Slovník	• Soupis slangových výrazů používaných v test casech a při komunikaci v rámci týmu

Zdroj: Vlastní zpracování

7.2 Vytvoření sdíleného dokumentu

- Slabé místo: jednotné testovací scénáře

Pro vyřešení problému s jednotnými testovacími scénáři potažmo test casy jsou jen dvě možná východiska. Prvním východiskem je úprava jednotlivých test casů test analytikem podle podmínek výrobce dané technické hry. Ačkoliv je to možné řešení, znamenalo by to značné navýšení práce s test casy pro test analytiky, tedy mohl by se tím vyskytnout problém na straně test analytiků, protože nebudou schopni produkovat takové množství test casů, jaké je vyžadováno.

Proto by bylo vhodnější zvolit druhou možnost, a sice vytvoření sdíleného dokumentu. Dokument by byl vytvořen na sdílené cloudové uložiště Google Disk, kde je již sepsáno několik podkladových materiálů, které jsou dostupné pro celé testovací oddělení.

Zmíněný dokument by obsahoval soupis všech výrobců technických her a jejich základní specifikace. Tabulka by byla vždy aktuální a kdokoli, po dohodě s kompetentní osobou, by jí mohl upravovat a případně psát poznámky. V následující tabulce je navržen formát, ve kterém by navrhovaný sdílený dokument mohl být vytvořen.

Tabulka 8 Návrh formátu sdíleného dokumentu

Jméno výrobce	SPECIFIKACE				
	Úvodní obrazovka	Výchozí nastavení zvuku	Automatická hra	Minimální vklad (Kč)	Maximální vklad (Kč)
Výrobce č. 1	ano	zapnuto	ano	5	1000
Výrobce č. 2	ne	vypnuto	ano	2,5	260
Výrobce č. 3	ano	vypnuto	ne	1	500

Zdroj: Vlastní zpracování

7.3 Nákup a přesun testovacích zařízení

- Slabé místo: nedostatek testovacích zařízení

S postupnou expanzí testovacího oddělní a s tím i testerů je nasnadě dokoupit testovací zařízení. Nejčastější překážkou práce je nedostatek mobilních zařízení iPhone. Proto navrhuji nakoupit alespoň dva nové iPhony a s postupující expanzí případně dokoupit i další zařízení. U nových iPhonů je důležité, aby byly stejného typu, jako jsou ty, kterými již testovací oddělní disponuje. Je to z toho důvodu, že nákup iPhonu, který ještě v portfoliu testovacího oddělení není, by způsobil to, že se budou muset vytvořit test casy na další typy mobilních zařízení a problém s nedostatkem iPhonů to neodstraní.
Dále navrhuji přesunout jeden terminál POS z jiné části budovy do kanceláře, kde sídlí testeři. Přítomnost terminálu POS přímo v kanceláři by velice urychlila vyřizování rychlých požadavků jako je třeba ověření uživatelského testovacího účtu.

7.4 One to one meeting

- Slabé místo: výkonnost testerů

Pro lepší výkonnost testerů navrhuji zavedení pravidelných one to one meetingů, které by spočívaly v diskusi Test coordinatora a jednoho testera.
Meetingy by se měly uskutečňovat s pravidelností jednou měsíčně, vždy v posledním týdnu v měsíci. U tohoto meetingu je důležité, aby probíhal strukturovaně. Test coordinator vždy rozešle testerům v minimálním předstihu jednoho dne otázky, které budou na meetingu položeny. Díky tomu si bude moct tester v klidu promyslet odpovědi,

případně si sepsat poznámky tak, aby mohlo být na samotném meetingu probráno všechno, co by tester chtěl probrat a na nic nezapomněl.
Samotný meeting by pak měl být rozdělený do dvou částí. Nejprve by Test coordinator zhodnotil výkonnost testera na základě statistik ze SpiraTest. Zhodnocení by mělo být spíše orientační, mělo by se spíše využít k pochvale testerů s nadprůměrnými výsledky, a jako podklad pro diskusi s těmi, kteří mají výsledky výrazně horší. Následně by položil připravené otázky, jako například:

- *Co Vás tento měsíc v práci nejvíce bavilo?*
- *Co Vám při práci vadilo nebo Vás při práci zdržovalo?*
- *Jak se vám pracovalo s novým členem týmu?*
- *Co by mohlo zvýšit Vaši produktivitu práce?*

V těchto otázkách by se měla vždy objevit i otázka na provedenou změnu nebo novinku, která v procesu testování předešlý měsíc nastala. Test coordinator tím získá zpětnou vazbu a bude vědět, zda mají provedené změny daného typu u testerů kladný ohlas nebo jsou pro ně zbytečné či dokonce jim ztěžují práci. Na konci výčtu otázek by měla zaznít otázka, zda má tester ještě nějaké připomínky či dotazy.
Na závěr test coordinator krátce shrne obsah meetingu. Meeting by měl končit pozitivně, aby tester odcházel motivován k dalším výkonům.

7.5 Informační tabule

- Slabé místo: nedostatečná informovanost a komunikace

Jelikož jsou všichni testeři zaměstnáni pouze na dohodu, chodí do práce průměrně dvakrát až třikrát týdně. To znamená, že nemohou být informovaní o všem, co bylo v předchozích dnech o testovaném objektu zjištěno či o jiných aktualitách.
Navrhuji tedy, aby se na nástěnnou popisovací tabuli, která již je v kanceláři testerů a není využitá, nadepsaly aktuálně testované releasy, kam by se psaly všechny důležité poznatky k testovanému releasu. Po jeho uzavření se poznámky smažou a bude nahrazen dalším releasem. Díky tomuto způsobu zaznamenávání budou mít testeři všechny novinky a poznámky k testovanému releasu vždy přehledně k dispozici. S každým ukončeným

releasem dostane jeden vybraný tester tabuli na starosti. Poznámky k již uzavřenému releasu smaže a nadepíše tabulku pro nový release.

8 Závěr

Digitalizace proniká do všech sfér života a rozšiřuje potřebu nejrůznějších profesí s ní spjatých. Někdy je vývoj až překotný a není dokumentován či uceleně popsán v odborné literatuře. Při zpracování této diplomové práce jsem tedy vycházela z dostupné odborné literatury, z internetových zdrojů a hlavně z osobních zkušeností nabytých při práci v oboru.

Testování softwaru je nedílnou součástí vývoje, která v poslední době nabývá na důležitosti ve značném počtu firem. Firmy si začínají uvědomovat, že čím dřív se defekt identifikuje, tím je jeho oprava méně nákladná. Z toho důvodu a také z důvodu postupující digitalizace, se testovací oddělení začínají rozrůstat, ale ne vždy je taková expanze pečlivě naplánována. Často se testovací oddělení rozrůstá velice nenápadně a bez nutné úpravy procesů a návazné změny organizační a řídící struktury.

Cílem diplomové práce bylo na základě popisu a rozboru současné praxe uživatelského testování v projektech ICT ve společnosti Sazka a.s. identifikovat slabá místa, nedostatky a navrhnout jejich možná řešení, a to v kontextu soudobé praxe.

Slabých míst současného stavu uživatelského testování, která přímo souvisí s prací testerů, bylo identifikováno celkem sedm. Tato slabá místa byla identifikována na základě popisu současného stavu testování, dále na základě diskuse s testery, kteří v Sazce pracují, a pomocí pozorování chodu testovacího oddělení. Ke každému slabému místu byl vytvořen návrh na jeho řešení a tím i zlepšení či zefektivnění některých procesů.

Stěžejním návrhem bylo zavedení důkladného školení testerů a vytvoření školicího manuálu, které by sloužilo nejen nově příchozím zaměstnancům, ale i stávajícím zaměstnancům, kteří školením neprošli. Součástí návrhu byl harmonogram školení i koncept školicího manuálu s rozpisem jednotlivých témat, která by měla být do manuálu zahrnuta. Tento návrh by řešil hned několik slabých míst najednou. Vyřešila by se tím absence školení nových testerů, nejednotná forma pro psaní incidentů, a to tím, že jedna kapitola ve školicím manuálu by byla věnována právě psaní incidentu a jeho formě, a navíc by takový školicí manuál mohl doplnit testovací plán, který v současné době není zcela kompletní. Neméně důležitým návrhem by bylo vytvoření sdíleného dokumentu se soupisem vybraných specifikací her jednotlivých výrobců, který by řešil problém s jednotnými testovacími scénáři potažmo test casy, jejichž nepřesnost přidělává testerům práci. Třetím návrhem je navýšení počtu testovacích zařízení nákupem alespoň dvou

iPhonů, a přesunem terminálu POS k testerům do kanceláře, čímž by se řešil nedostatek testovacího zařízení. Dalším návrhem podstatným pro zvýšení výkonnosti testerů je one to one meeting, který by měl zvýšit motivaci testerů k vyššímu pracovnímu nasazení a lepším výkonům. Posledním návrhem je zavedení informační tabule v kanceláři testerů, která by sloužila k předávání informací k právě probíhajícímu releasu i změnám a novinkám, které proběhly. Tím by se vyřešila nedostatečná komunikace mezi testery a jejich informovanost.

Diplomová práce se sice zabývala testovacím oddělením z pohledu testera, ale bylo by vhodné identifikovat slabá místa napříč celým testovacím oddělením a díky tomu vytvořit komplexní návrh řešení, které by mohlo mít pro chod testovacího oddělení pozitivní vliv jak z hlediska efektivity práce, tak z hlediska finančních úspor a spokojenosti zaměstnanců.

9 Seznam použitých zdrojů

Knižní zdroje:

BUREŠ, Miroslav, Miroslav RENDA, Michal DOLEŽEL, Peter SVOBODA, Zdeněk GRÖSSL, Martin KOMÁREK, Ondřej MACEK a Radoslav MLYNÁŘ. *Efektivní testování softwaru: klíčové otázky pro efektivitu testovacího procesu*. Praha: Grada, 2016. Profesionál. ISBN 978-80-247-5594-6.

MYERS, Glenford J., Corey SANDLER a Tom BADGETT. *The art of software testing*. 3rd ed. Hoboken, N.J.: John Wiley, c2012. ISBN 1118031962.

PAGE, Alan, Ken JOHNSTON a Bj ROLLISON. *Jak testuje software Microsoft*. Brno: Computer Press, 2009. ISBN 978-80-251-2869-5.

PATTON, Ron a Anna HAVLÍČKOVÁ. *Testování softwaru: průvodce testováním*. Praha: Computer Press, 2002. Programování. ISBN 80-722-6636-5.

ROUDENSKÝ, Petr a Anna HAVLÍČKOVÁ. *Řízení kvality softwaru: průvodce testováním*. Brno: Computer Press, 2013. ISBN 978-80-251-3816-8.

SCHOTANUS, Chris C. *TestFrame: an approach to structured testing*. Dordrecht: Springer, c2009. ISBN 978-3-642-00821-4.

Elektronické zdroje:

Automatizované testování. *Testování Softwaru* [online]. 2020 [cit. 2020-02-09]. Dostupné z: http://testovanisoftwaru.cz/automatizovane-testovani/

Bezpečnost a důvěryhodnost. *Sazka* [online]. 2020 [cit. 2020-03-13]. Dostupné z: https://www.sazka.cz/sazka-svet/o-spolecnosti/certifikaty

ISTQB Glossary: Slovník pojmů ISTQB [online]. 2018 [cit. 2020-01-06]. Dostupné z: http://castb.org/wp-content/uploads/2018/09/ISTQB_CZ_Glossary_20180831.pdf

Management. *Sazka* [online]. 2020 [cit. 2020-03-13]. Dostupné z: https://www.sazka.cz/sazka-svet/o-spolecnosti/management

PRABHU, Daniel. Bug Life Cycle. *Software Testing Daniel* [online]. 11. 3. 2015 [cit. 2020-03-20]. Dostupné z: https://softwaretestingdaniel.wordpress.com/2015/03/11/bug-life-cycle/

REQTEST. A quick guide to Test Scenario and Test Case. ReQtest [online]. 19. 9. 2018 [cit. 2020-01-23]. Dostupné z: https://reqtest.com/testing-blog/test-scenario-test-case/

Sample Test Case Template With Test Case Examples. Software Testing Help [online]. 10. 11. 2019 [cit. 2020-01-23]. Dostupné z: https://www.softwaretestinghelp.com/test-case-template-examples/

SAZKA A.S. *Výroční zpráva SAZKA a.s. 2018* [online]. Praha 9, 2019 [cit. 2020-01-30]. Dostupné z: https://www.sazka.cz/SazkaWeb/media/content/Vyrocni-zprava-SAZKA-a-s-2018.pdf?ext=.pdf

Sazka Olympijská cesta. *Sazka* [online]. 2020 [cit. 2020-03-13]. Dostupné z: https://www.sazka.cz/sazka-svet/o-spolecnosti/sazka-olympijska-cesta

STC admin. Software Testing Life Cycle STLC. *Software Testing Class* [online]. 2. 10. 2013 [cit. 2020-02-27]. Dostupné z: https://www.softwaretestingclass.com/software-testing-life-cycle-stlc/

STC admin. What is difference between Test Cases vs Test Scenarios? *Software Testing Class* [online]. 24. 10. 2014 [cit. 2020-03-05]. Dostupné z: https://www.softwaretestingclass.com/what-is-difference-between-test-cases-vs-test-scenarios/

Stručná historie. *Sazka* [online]. 2019 [cit. 2020-03-13]. Dostupné z: https://www.sazka.cz/sazka-svet/o-spolecnosti/historie

Testovací dokumentace – plán, scénář, případ. *Testování Software* [online]. 2020 [cit. 2020-03-02]. Dostupné z: http://test.swtestovani.cz/index.php?option=com_content&view=article&id=15:testovaci-dokumentace-plan-scena-pipad&catid=3:zaklady&Itemid=11

The Waterfall Model. *Free management books* [online]. 2020 [cit. 2020-03-11]. Dostupné z: http://www.free-management-ebooks.com/news/waterfall-model/

Vodopádový model. *Testování Softwaru* [online]. 2020 [cit. 2020-02-20]. Dostupné z: http://testovanisoftwaru.cz/manualni-testovani/modely-zivotniho-cyklu-softwaru/vodopadovy-model/

What is V-model and advantages and disadvantages of this model. *Software Testing* [online]. 24. 6. 2015 [cit. 2020-03-20]. Dostupné z: https://testingsorttricks.wordpress.com/2015/07/24/what-is-v-model-and-advantages-and-disadvantages-of-this-model/